HAPPH
Servicios de hotel en el móvil

Juan Carlos Feliu Ladaria

Carlos Gutiérrez Hernández-Gil

Jacobo Olmedo Martín

Javier Pavón Núñez

Trabajo Fin de Grado

Facultad de Informática
Universidad Complutense de Madrid

Curso 2013–2014

Profesor:
Juan Pavón Mestras

HAPPH
Servicios de hotel en el móvil

Trabajo Fin de Grado

Facultad de Informática

Universidad Complutense de Madrid

Autores:

Juan Carlos Feliu Ladaria

Carlos Gutiérrez Hernández-Gil

Jacobo Olmedo Martín

Javier Pavón Núñez

Profesor director:
Juan Pavón Mestras

Curso 2013-2014

Autorizamos a la Universidad Complutense de Madrid a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, la documentación y/o el prototipo desarrollado.

En Madrid, a 6 de junio de 2014.

Juan Carlos Feliu Ladaria

Carlos Gutiérrez Hernández-Gil

Jacobo Olmedo Martín

Javier Pavón Núñez

A todos nuestros familiares
y amigos, por su apoyo incondicional
y confiar en nuestras capacidades

AGRADECIMIENTOS

Gracias a nuestro profesor D. Juan Pavón Mestras por su disposición a aportar nuevas ideas, consejos, enseñanzas y paciencia durante la elaboración de este proyecto.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

PRÓLOGO

Por fin, y tras un mes mirando viajes, ya estás ahí. Has mirado lugares que visitar, restaurantes donde comer, e incluso te has comprado un libro con frases típicas para poder comunicarte con la gente, sabiendo que al final utilizarás el inglés, como es costumbre cuando viajas a otro país. El problema es que de inglés sabes lo justo y encima te da vergüenza hablarlo con desconocidos.

Por fin llegaste a la entrada del hotel donde vas a vivir los próximos días. Intentas comunicarte como buenamente puedes con el empleado de hotel, que casualmente habla cuatro idiomas pero el español no es uno de ellos. Después de una torpe comunicación con el empleado, has entendido que número de habitación tienes, y además te da un papel con un nombre de usuario y una contraseña. Creías que era para el Wifi del hotel, sin embargo el empleado te indica que te bajes una aplicación Android para el móvil.

Dejando las maletas en la habitación, te entra la curiosidad y decides descargarte la aplicación para ver de lo que hablaba el empleado. La instalas y la ejecutas. -¿HappH?¿Qué es esto que me ha mandado descargar el del hotel?- te preguntas, mientras decides poner el nombre de usuario y la contraseña que te han facilitado cuando has ido a recoger las llaves de la habitación.

Empezaste a navegar por la aplicación hasta que te diste cuenta de lo que era. No entendiste mucho de lo que te decía el señor, sin embargo ahora empezaba a cobrar sentido.

Una de las cosas que te daba vergüenza era la de tener que llamar por teléfono al lobby del hotel preguntando por servicios del hotel, por platos de comidas, etcétera. ¡Ahora eso se acabó!, tienes en tus manos una aplicación que puede hacer las peticiones por ti, sin necesidad de tener que ponerte en contacto con los del hotel, de estar media hora intentando comprender lo que te dicen y lo que les dices a ellos.

Además, con esta aplicación es bastante fácil, ya que puedes disfrutar de los servicios del hotel sin apenas esfuerzo.

HappH es una aplicación que te hace la vida más fácil dentro del hotel, para que lo único en lo que tengas que pensar sea en disfrutar.

RESUMEN

HappH es una aplicación hotelera multiplataforma cuyo objetivo es el de facilitar a los clientes de un hotel la petición de servicios que éste ofrece de forma sencilla, rápida e intuitiva, ya que no se requieren conocimientos por parte del cliente para poder disponer de toda su funcionalidad.

HappH está compuesta por una aplicación para dispositivos Android diseñada especialmente para el cliente, desde la cual una persona puede solicitar los servicios ofrecidos por el hotel, además de consultar el estado en el que se encuentran los servicios que haya solicitado, anularlos o acceder a información de interés que el hotel haya facilitado.

Además, HappH posee una aplicación web donde los clientes podrán realizar las mismas operaciones que desde la aplicación Android, mientras que el personal del hotel podrá consultar todos los servicios que los clientes hayan solicitado, organizarlos y tramitarlos.

Tanto la plataforma Android como la web están conectadas, ya que la persona responsable de gestionar la aplicación Android podrá configurarla desde la web, configurando y modificando los servicios que desea ofrecer, el nombre de los mismos, su precio, los empleados que pueden acceder a ella, etcétera.

Palabras clave:

- Aplicación
- Hotel
- Servicios
- Android
- App
- Ingeniería software

ABSTRACT

HappH is a hotel application multiplatform whose aim is to help customers requesting for services that the hotel provides in a simple, fast and intuitive way, since knowledge is not required by the client to have full functionality required.

HappH consists of an application for Android devices specially designed for the client, from which a person may request the services the hotel provides them, in addition to consulting the state which are the services requested, cancel or access information of interest that the hotel provides.

In addition, HappH also has a web application where customers can perform the same operations from the Android application, while the staff can check all the services that customers have requested, organize and process them.

Both platforms are connected, as the person responsible can configure the Android application from the web, configuring and modifying the services that the hotel wants to offer to their clients, rename them, change its price, etc.

Keywords:

- Application
- Hotel
- Services
- Android
- App
- Software Engineering

1. INTRODUCCIÓN

El sistema Android ha sido una de las tecnologías más recientes que ha salido al mercado. A día de hoy, prácticamente todo el mundo dispone de un Smartphone, la cifra de estos dispositivos superan los 1700 millones en todo el planeta.

Estos móviles permiten disponer de múltiples funcionalidades para los usuarios, e incluso permite automatizar servicios que antes no lo estaban. De este hecho nace la idea de poder automatizar la gestión de todos los servicios que puede prestar un hotel a sus clientes. Hasta ahora, para poder pedir un servicio, había que llamar a recepción para poder solicitarlo, pero con HappH esta tarea se puede realizar de forma fácil y sencilla, permitiendo también al hotel disponer de una organización más eficiente de las peticiones de sus clientes.

HappH dispone de una interfaz ágil, sencilla e intuitiva que permite al usuario manejar todos los servicios disponibles de forma clara y precisa.

¿Qué nos ha llevado a realizar este proyecto? Como hemos citado anteriormente, el crecimiento de Smartphones está siendo exponencial, y poco a poco se está convirtiendo en un elemento indispensable para nuestras vidas. Esto último junto al sector del turismo, que cobra vital importancia en numerosos países, y más aún en España, hacen pensar en un producto orientado a este sector, ya que el éxito que podría tener (por las facilidades que presta para aquellas personas que se alojen en hoteles), hace que este proyecto pueda ser viable.

En cuanto a los objetivos de este proyecto, la idea que ha tenido este grupo de trabajo es la de poder crear una aplicación personalizable para cada hotel, ampliando así la cuota de mercado en la que se podría manejar.

Esta plataforma no pretende sustituir la operativa de entradas y salidas de un hotel, sino que funciona como una parte auxiliar, ya que no se encarga de la gestión de las reservas o de los clientes.

Por último, dado que el sector del turismo es uno de los sectores que más beneficios puede producir, unido a la tecnología que usamos en nuestro día a día asegura que este proyecto además de viable pueda tener futuro en el mercado.

2. POSICIONAMIENTO

Actualmente existen aplicaciones parecidas a la que se ha desarrollado, como por ejemplo “PlugHotel”, una aplicación que ayuda a pedir servicios durante la estancia en el hotel. Sin embargo, esta aplicación no es muy genérica, en cambio HappH es una aplicación constituida a partir de plantillas de servicios para que el cliente que quiera esta aplicación, tan solo comunicándolo, se pueda personalizar en un espacio de tiempo relativamente breve.

Otra aplicación similar es “Loungeup”, aplicación de servicio personalizado para usuarios de hoteles. Esta aplicación sí tiene mayores similitudes con la aplicación desarrollada en este proyecto.

Lo que nuestra aplicación ofrece a un hotel es la completa personalización del mismo. De esta forma, no hay dos aplicaciones HappH que sean iguales, ya que cada uno dispondrá de diferentes servicios de un hotel, además de la posibilidad de añadir los colores corporativos del hotel en cuestión para darle más personalización aún.

La idea de esta aplicación es la de facilitar tanto a los usuarios de un hotel como a los empleados a poder organizar las peticiones hechas por los clientes de una forma efectiva, permitiendo que el servicio que presta el hotel sea mucho más rápido y eficaz, sin necesidad de llamadas, tan solo presionando en los botones de la aplicación.

Además, dado que la aplicación es fácil de utilizar y no requiere de conocimiento alguno por parte del cliente sobre cómo funciona un hotel, hace que esta aplicación sea muy intuitiva y que en pocos pasos se pueda solicitar el servicio que desea el cliente.

PlugHotel

Es una aplicación más compleja que HappH, que ofrece información, servicios del hotel y alrededores donde el huésped se aloja.

Opcionalmente, esta aplicación podrá integrarse mediante un API con el PMS del hotel, de esta forma el PMS podrá enviar los datos del huésped para que la aplicación pueda ofrecer servicios e información personalizada en base a la estancia del huésped. Esta aplicación podrá ofrecer:

Información del Hotel y Oferta

Información de Eventos a destacar (personalizados o no dependiendo de la integración con el PMS)

Información de Oferta complementaria (excursiones, restaurantes etc.)

Servicios de "MarketPlace" del Hotel (venta de productos y servicios, "merchandising"...)

Posibilidad de que el huésped pueda realizar cambios y solicitar servicios asociados a la estancia, solicitar los datos de su reserva, factura y realizar el pago de esta a través del móvil.

El hotelero con esta herramienta puede ofrecer información antes, durante y después de la estancia del Huésped (fidelización)

http://www.plughotel.com/

LoungeUp

Esta aplicación permite a los hoteleros crear un portal de bienvenida en el móvil, la tableta o el ordenador portátil de sus clientes. Esta aplicación facilita la estancia de los clientes permitiéndoles pedir platos a la habitación o reservar una sesión de spa sin necesidad de pasar por recepción.

Como vemos, LoungeUp es una aplicación muy parecida a HappH, sin embargo, al estar desarrollada mediante plantillas, HappH permite implementar servicios de forma fácil y sencilla para el programador, haciendo que esta además de ser personalizable, posiblemente se tarde menos tiempo en realizar dichas implementaciones de personalización que en LoungeUp.

http://es.kioskea.net/faq/9090-loungeup-aplicacion-de-servicio-personalizado-para-usuarios-de-hoteles

Conclusiones

Aunque no disponemos de las líneas de programación de estas dos aplicaciones citadas, HappH posee un dinamismo que las anteriores posiblemente no tengan, ya que sus desplegables, opciones y menús se pueden cambiar de una forma rápida y sencilla. Además, esta aplicación permite poner los colores corporativos del hotel en la aplicación, haciendo que esta se pueda distinguir de otros hoteles que utilicen la misma aplicación, permitiendo que tenga un efecto de exclusividad frente al cliente que pide nuestra aplicación.

Además, al contrario que las otras, HappH dispone de un acceso web para no estar obligado a tener un teléfono Android o tener que descargarte la aplicación para poder utilizar los servicios de forma sencilla, sino que a través de la web puedes hacer lo mismo que haces en la aplicación. Otro de los puntos fuertes de HappH frente a las otras dos aplicaciones comentadas, es que permite que incluso se puedan cancelar servicios o peticiones en tiempo real, con tan solo unos clics en el caso del administrador que desee cancelar un servicio, o pulsando el servicio en la aplicación en el caso del cliente para

poder cancelarlo. Es por eso que HappH puede llegar a ser una dura competidora frente al resto de aplicaciones que tienen una funcionalidad parecida.

3. ESPECIFICACIÓN DE REQUISITOS

3.1. Introducción

HaapH se centra en la gestión de peticiones de servicio ofrecidos por un hotel desde terminales móviles Android o desde un navegador web instalado en un ordenador personal.

La funcionalidad de los servicios ofrecidos se expresa a partir de casos de uso y escenarios que reflejan los objetivos planeados del proyecto. En primer lugar se fijan y definen con detalle los requisitos que se consideran fundamentales para después, una vez implementados y funcionando correctamente, poderse apoyar en ellos para realizar ampliaciones futuras si fuera necesario.

A continuación se muestran los diagramas de casos de uso general, los cuales muestran con claridad la funcionalidad inicial que se le quiere dar a la aplicación. Posteriormente se entra en el detalle de cada caso de uso de forma que se refleje el comportamiento del sistema para facilitar su construcción e implementación posterior.

Por último señalar que todos los diagramas UML del presente documento han sido diseñados con la herramienta Enterprise Architect, la cual permite trabajar de manera sencilla con este tipo de diagramas.

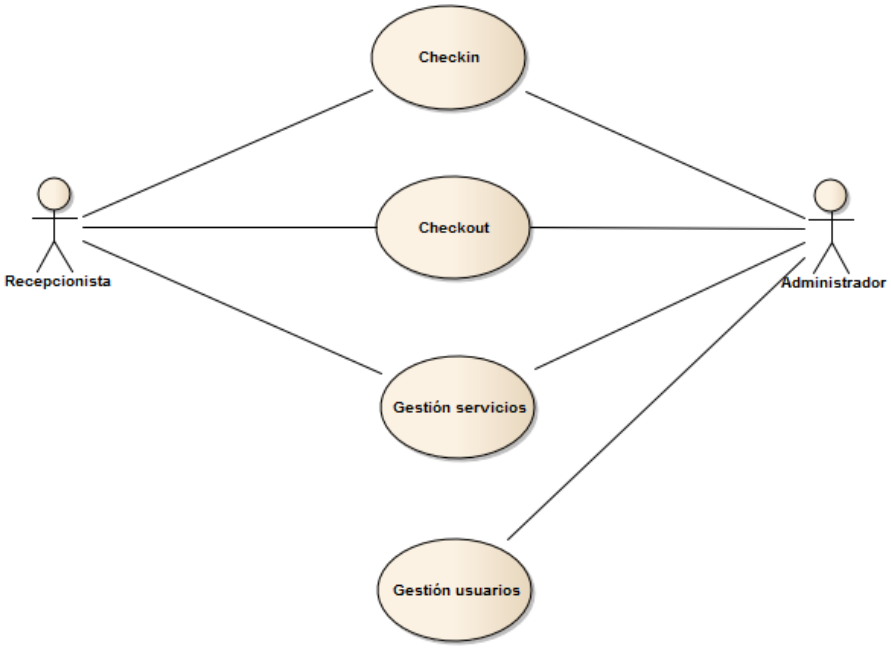


Ilustración 1 .Diagrama de casos de uso para empleados

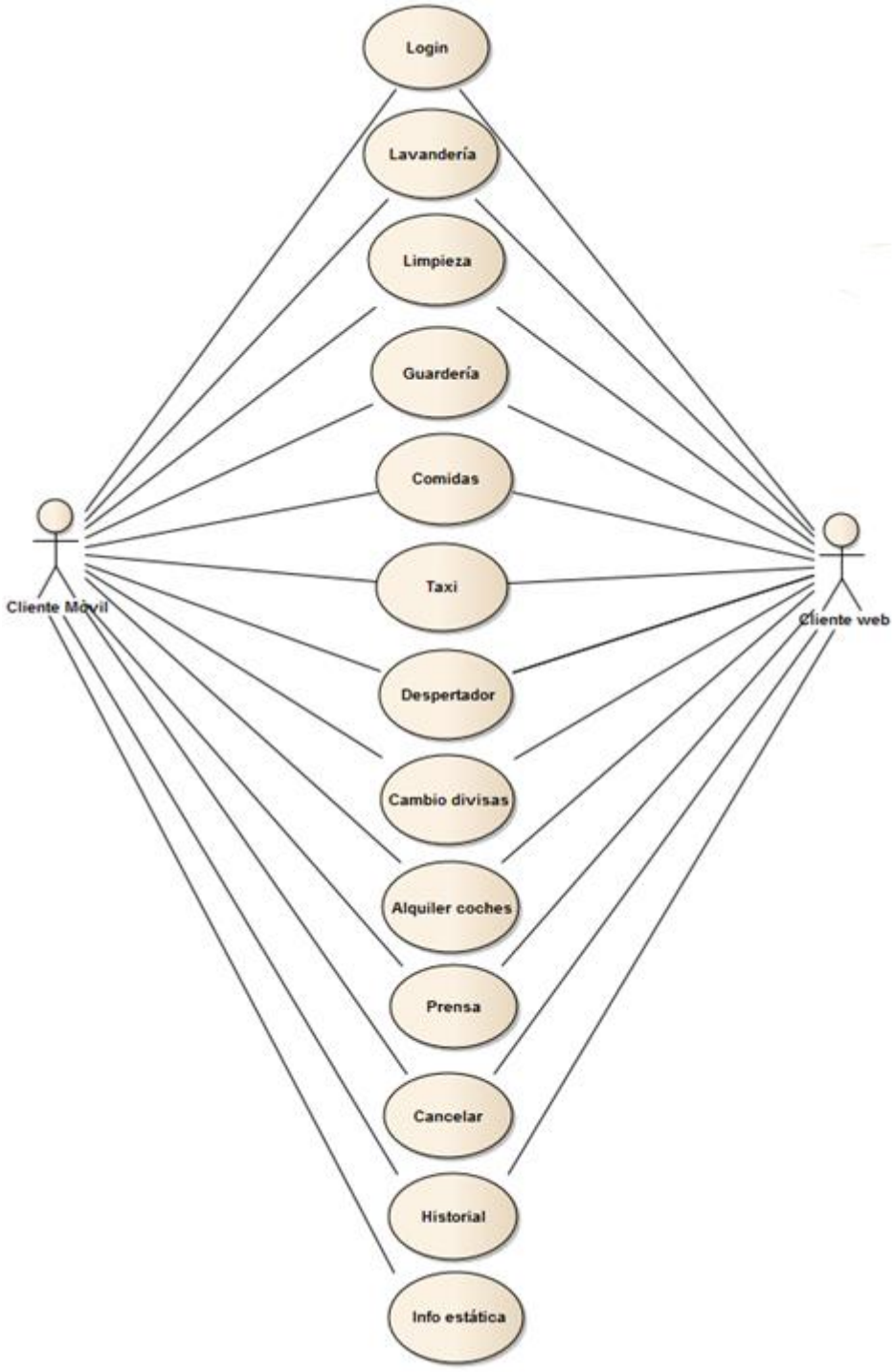


Ilustración 2. Diagrama de casos de uso para clientes.

3.2. Requisitos funcionales

3.2.1. Login

Una vez iniciada la aplicación móvil o la aplicación web se muestra la pantalla de autenticación, en la cual el cliente o empleado deberá identificarse para poder acceder al área privada.

CASO DE USO	Login
Objetivo en contexto	Es la pantalla inicial donde clientes y empleados del hotel tendrán que teclear su nombre de usuario y su contraseña para poder acceder a la aplicación.
Entradas	Nombre de usuario, contraseña.
Precondiciones	El usuario no tiene establecida una sesión.
Salidas	-
Postcondición si éxito	El sistema proporciona la pantalla principal de la aplicación. Se establece una sesión para el usuario.
Postcondición si fallo	Devolverá un mensaje del servidor indicando que la contraseña o el usuario no son correctos, o que ha habido un error al conectar.
Actores	Cliente, empleados del hotel.

Secuencia normal	**Paso**	**Acción**
	1	Desde la pantalla de login de la aplicación, el cliente o el empleado del hotel deberán escribir su nombre de usuario y la contraseña proporcionados por el hotel. De forma opcional, el usuario podrá pulsar sobre recordar contraseña para que la próxima vez que utilice la aplicación no tenga que volver a poner el nombre de usuario y la contraseña.
	2	El cliente proporciona nombre de usuario y contraseña, y pulsa sobre el botón aceptar para enviar la petición. Si hubiese un problema al enviar la petición, ir a E1.
	3	El servidor recibe la petición y valida los datos. Si los

		datos no son correctos, ir a E2.
	4	Si los datos son correctos, el sistema muestra la pantalla de bienvenida.
Secuencias alternativas	**Paso**	**Acción**
	E1	La aplicación muestra un mensaje: "Imposible de conectar, inténtelo más tarde".
	E2	El servidor envía un mensaje de datos incorrectos y la aplicación muestra "Usuario o contraseña incorrectos".

3.2.2. Servicio de Taxi

Con esta funcionalidad, los clientes podrán solicitar al hotel que un taxi les recoja en una fecha y hora determinada.

CASO DE USO	Servicio de taxi
Objetivo en contexto	Permite al usuario informar al hotel de que necesita que llamen a un taxi. Los empleados del hotel harán los trámites necesarios.
Entradas	-
Precondiciones	El cliente debe estar logueado en la aplicación.
Salidas	-
Postcondición si éxito	Si el servicio se ha solicitado correctamente, se mostrará un mensaje indicando que la operación ha sido un éxito y volverá a la pantalla de inicio.
Postcondición si fallo	Si hubiese un fallo, se mostrará un mensaje de error indicando los motivos del error y volverá a la pantalla de inicio
Actores	Cliente

Secuencia normal	**Paso**	**Acción**

	1	El cliente selecciona el servicio de taxi
	2	El cliente elegirá la fecha y la hora en la que desea tener el taxi.
	3	Si se confirma la petición, se crea el nuevo servicio y se notifica en la pantalla del móvil que se ha creado correctamente, informando de los días, horas y minutos que faltan hasta la fecha y hora solicitada. Si no se ha podido crear, ir a E1.
	4	Si el cliente cancela la operación (botón cancelar), volver a menú de la aplicación.
Secuencias alternativas	**Paso**	**Acción**
	E1	Se mostrará un mensaje informativo indicando que hubo un problema y volverá a la pantalla de inicio.

3.2.3. Servicio de Limpieza

El servicio de limpieza permite notificar al hotel por parte del cliente la necesidad de que le limpien la habitación en una determinada hora.

CASO DE USO	Servicio de limpieza	
Objetivo en contexto	Solicitar el servicio de limpieza de habitación a una hora determinada	
Entradas	-	
Precondiciones	El cliente debe estar logueado en la aplicación.	
Salidas	-	
Postcondición si éxito	Si el servicio se ha solicitado correctamente, se mostrará un mensaje indicando que la operación ha sido un éxito y volverá a la pantalla de inicio.	
Postcondición si fallo	Si hubiese un fallo, se mostrará un mensaje de error indicando los motivos del error y volverá a la pantalla de inicio	
Actores	Cliente	
Secuencia	**Paso**	**Acción**

normal		
	1	El cliente selecciona el servicio de limpieza.
	2	El cliente elegirá la hora a la que desea que le limpien la habitación.
	3	Si se confirma el servicio, se crea el nuevo servicio y se notifica en la pantalla del móvil que se ha creado correctamente. Si no se ha podido crear, ir a E1.
	4	Si el cliente cancela la operación (botón cancelar), volver a menú de la aplicación.
Secuencias alternativas	**Paso**	**Acción**
	E1	Se mostrará un mensaje informativo indicando que hubo un problema y volverá a la pantalla de inicio.

3.2.4. Servicio de Lavandería

El servicio de lavandería facilita al cliente solicitar al hotel la limpieza y planchado de unas determinadas prendas y fijar una hora para su recogida por parte de los empleados.

CASO DE USO	Servicio de lavandería
Objetivo en contexto	Solicitar la limpieza de unas determinadas prendas.
Entradas	-
Precondiciones	El cliente debe estar logueado en la aplicación.
Salidas	-
Postcondición si éxito	Si el servicio se ha solicitado correctamente, se mostrará un mensaje indicando que la operación ha sido un éxito y volverá a la pantalla de inicio.
Postcondición si fallo	Si hubiese un fallo, se mostrará un mensaje de error indicando los motivos del error y volverá a la pantalla de inicio

Actores	Cliente

Secuencia normal	**Paso**	**Acción**
	1	El cliente selecciona el servicio de lavandería.
	2	Se muestra un desplegable con el tipo de ropa que se desea lavar, y un campo para seleccionar la cantidad.
	3	Al seleccionar el tipo de prenda y la cantidad, se añade a una lista que formará el pedido, pudiendo elegir más prendas.
	4	Al terminar de elegir todo aquello que se desea lavar, se selecciona la hora de recogida de la ropa por parte del hotel y se confirma el servicio pulsando el botón “Enviar pedido”.
	5	Si se confirma el pedido, se crea el nuevo servicio y se notifica en la pantalla del móvil que se ha creado correctamente, notificando también el tiempo en días, horas y minutos que queda para que se recoja la ropa. Si no se ha podido crear, ir a E1.
	6	Si el cliente cancela la operación (botón cancelar), volver a menú de la aplicación
Secuencias alternativas	**Paso**	**Acción**
	E1	Se mostrará un mensaje informativo indicando que hubo un problema y volverá a la pantalla de inicio.

3.2.5. Servicio Despertador

Con esta funcionalidad el cliente podrá fijar la hora a la que desea ser despertado telefónicamente.

CASO DE USO	Servicio despertador

Objetivo en contexto	Solicitar el servicio de despertador. Se podrá solicitar para un día en concreto a una hora determinada, o para recibirlo durante toda la estancia a la misma hora.
Entradas	-
Precondiciones	El cliente debe estar logueado en la aplicación.
Salidas	-
Postcondición si éxito	Si el servicio se ha solicitado correctamente, se mostrará un mensaje indicando que la operación ha sido un éxito y volverá a la pantalla de inicio.
Postcondición si fallo	Si hubiese un fallo, se mostrará un mensaje de error indicando los motivos del error y volverá a la pantalla de inicio
Actores	Cliente

Secuencia normal	**Paso**	**Acción**
	1	El cliente selecciona el servicio despertador.
	2	Selecciona la hora del servicio, pudiendo marcar la opción de recibirlo todos los días que dure la estancia.
	3	Si se confirma la petición, se crea el nuevo servicio y se notifica en la pantalla del móvil que se ha creado correctamente. Si no se ha podido crear, ir a E1.
	4	Si el cliente cancela la operación (botón cancelar), volver a menú de la aplicación
Secuencias alternativas	**Paso**	**Acción**
	E1	Se mostrará un mensaje informativo indicando que hubo un problema y volverá a la pantalla de inicio.

3.2.6. Servicio de Guardería

El servicio de guardería permite al cliente solicitar que le recojan y cuiden a sus hijos en una fecha y hora determinada.

CASO DE USO	Servicio de guardería

Objetivo en contexto	Solicitar el servicio de guardería a una hora determinada para un número concreto de niños.
Entradas	-
Precondiciones	El cliente debe estar logueado en la aplicación.
Salidas	-
Postcondición si éxito	Si el servicio se ha solicitado correctamente, se mostrará un mensaje indicando que la operación ha sido un éxito y volverá a la pantalla de inicio.
Postcondición si fallo	Si hubiese un fallo, se mostrará un mensaje de error indicando los motivos del error y volverá a la pantalla de inicio
Actores	Cliente

Secuencia normal	**Paso**	**Acción**
	1	El cliente selecciona el servicio de guardería.
	2	Selecciona la hora para el servicio y el número de niños.
	4	Si se confirma la petición, se crea el nuevo servicio y se notifica en la pantalla del móvil que se ha creado correctamente. Si no se ha podido crear, ir a E1.
	5	Si el cliente cancela la operación (botón cancelar), volver a menú de la aplicación
Secuencias alternativas	**Paso**	**Acción**
	E1	Se mostrará un mensaje informativo indicando que hubo un problema y volverá a la pantalla de inicio.

3.2.7. Servicio de Comidas

A través del servicio de comidas el cliente podrá elegir entre una serie de platos que ofrece el hotel y solicitar que se lo suban a la habitación.

CASO DE USO	Servicio de comidas

Objetivo en contexto	Solicitar un menú de una lista de platos ofertados	
Entradas	-	
Precondiciones	El cliente debe estar logueado en la aplicación.	
Salidas	-	
Postcondición si éxito	Si el servicio se ha solicitado correctamente, se mostrará un mensaje indicando que la operación ha sido un éxito y volverá a la pantalla de inicio.	
Postcondición si fallo	Si hubiese un fallo, se mostrará un mensaje de error indicando los motivos del error y volverá a la pantalla de inicio	
Actores	Cliente	
Secuencia normal	**Paso**	**Acción**
	1	El cliente selecciona el servicio de comidas, y navega por las distintas categorías del menú hasta encontrar la categoría correspondiente
	2	Una vez elegida la categoría, se muestran los diferentes platos y el importe de cada uno de ellos.
	3	Al elegir un plato, se accede al menú para elegir la cantidad que se desea recibir del mismo y dos botones para aceptar el servicio o volver atrás.
	4	Si se confirma el menú, se crea el nuevo servicio y se notifica en la pantalla del móvil que se ha creado correctamente. Si no se ha podido crear, ir a E1.
	5	Si el cliente cancela la operación (botón cancelar), volver a menú de la aplicación
Secuencias alternativas	**Paso**	**Acción**
	E1	Se mostrará un mensaje informativo indicando que hubo un problema y volverá a la pantalla de inicio.

3.2.8. Servicio de Prensa

Con este servicio, el usuario podrá elegir su periódico o revista preferida y solicitar que se lo entregan en la habitación en la hora indicada, para un día puntual o durante toda su estancia en el hotel.

CASO DE USO	Servicio de prensa
Objetivo en contexto	Solicitar un periódico de una lista de periódicos ofertados. Se podrá solicitar para un día en concreto a una hora determinada, o para recibirlo durante toda la estancia a la misma hora.
Entradas	-
Precondiciones	El cliente debe estar logueado en la aplicación.
Salidas	-
Postcondición si éxito	Si el servicio se ha solicitado correctamente, se mostrará un mensaje indicando que la operación ha sido un éxito y volverá a la pantalla de inicio.
Postcondición si fallo	Si hubiese un fallo, se mostrará un mensaje de error indicando los motivos del error y volverá a la pantalla de inicio
Actores	Cliente

Secuencia normal	**Paso**	**Acción**
	1	El cliente selecciona el servicio de prensa.
	2	Selecciona el periódico que desea y la hora de entrega del periódico, pudiendo marcar la opción de recibirlo todos los días que dure la estancia.
	3	Si se confirma el periódico, se crea el nuevo servicio y se notifica en la pantalla del móvil que se ha creado correctamente. Si no se ha podido crear, ir a E1.
	4	Si el cliente cancela la operación (botón cancelar), volver a menú de la aplicación
Secuencias alternativas	**Paso**	**Acción**
	E1	Se mostrará un mensaje informativo indicando que hubo un problema y volverá a la pantalla de inicio.

3.2.9. Servicio de Cambio de Divisas

El hotel ofrece un servicio de cambio de divisas que el cliente puede solicitar desde su móvil, indicando las divisas, viendo la tasa de cambio y el importe final de la operación. Cuando la moneda esté disponible será informado por recepción.

CASO DE USO	Servicio de cambio de divisas	
Objetivo en contexto	Funcionalidad que permite al usuario pedir dinero en efectivo en la moneda que desee (las que proporcione el hotel). Inmediatamente habrá un cargo en la cuenta del cliente por el valor del dinero que ha pedido en efectivo.	
Entradas	-	
Precondiciones	El cliente debe estar logueado en la aplicación.	
Salidas	-	
Postcondición si éxito	Si el servicio se ha solicitado correctamente, se mostrará un mensaje indicando que la operación ha sido un éxito y volverá a la pantalla de inicio.	
Postcondición si fallo	Si hubiese un fallo, se mostrará un mensaje de error indicando los motivos del error y volverá a la pantalla de inicio	
Secuencia normal	**Paso**	**Acción**
	1	El cliente selecciona el servicio de cambio de divisas.
	2	Introduce la cantidad a cambiar y selecciona el cambio de monedas, pudiendo ver por pantalla la cantidad a recibir.
	4	Si se confirma el cambio, se crea el nuevo servicio y se notifica en la pantalla del móvil que se ha creado correctamente. Si no se ha podido crear, ir a E1.
	5	Si el cliente cancela la operación (botón cancelar), volver a menú de la aplicación
Secuencias alternativas	**Paso**	**Acción**
	E1	Se mostrará un mensaje informativo indicando que hubo un problema y volverá a la pantalla de inicio.

3.2.10. Servicio de Alquiler de Coches

Mediante esta opción, los clientes pueden solicitar el alquiler de un coche eligiendo la gama y modelo y el número de días que lo utilizará.

CASO DE USO	Servicio de alquiler de coches
Objetivo en contexto	Solicitar el alquiler de un coche de una lista de coches ofertados
Entradas	-
Precondiciones	El cliente debe estar logueado en la aplicación.
Salidas	-
Postcondición si éxito	Si el servicio se ha solicitado correctamente, se mostrará un mensaje indicando que la operación ha sido un éxito y volverá a la pantalla de inicio.
Postcondición si fallo	Si hubiese un fallo, se mostrará un mensaje de error indicando los motivos del error y volverá a la pantalla de inicio
Actores	Cliente

Secuencia normal	**Paso**	**Acción**
	1	El cliente selecciona el servicio de alquiler de coches, y navega por los distintos tipos de coches (Pequeño, Mediano, Familiar…) hasta encontrar el tipo correspondiente
	2	Una vez elegido el tipo, se muestran los diferentes coches y el importe de alquiler de cada uno de ellos (por día).
	3	Al elegir un coche, se accede al menú para elegir el coche que se desea alquilar, un selector para el número de días y dos botones para aceptar el servicio o volver atrás. Además se muestra el importe total dependiendo del número de días solicitados.
	4	Si se confirma el menú, se crea el nuevo servicio y se notifica en la pantalla del móvil que se ha creado correctamente. Si no se ha podido crear, ir a E1.

	5	Si el cliente cancela la operación (botón cancelar), volver a menú de la aplicación
Secuencias alternativas	**Paso**	**Acción**
	E1	Se mostrará un mensaje informativo indicando que hubo un problema y volverá a la pantalla de inicio.

3.2.11. Cancelar un servicio y ver Historial

Con esta opción el cliente podrá ver el histórico de servicios solicitados durante su estancia y solicitar la cancelación de aquellos servicios que estén pendientes de tramitar.

CASO DE USO	Historial
Objetivo en contexto	Ver el historial de los servicios solicitados y su estado, pudiendo cancelar aquellos servicios que están pendientes de ser tramitados.
Entradas	-
Precondiciones	El cliente debe estar logueado en la aplicación.
Salidas	-
Postcondición si éxito	En el caso de anular un servicio, si el servicio se ha anulado correctamente, se mostrará un mensaje indicando que la operación ha sido un éxito y volverá a mostrar el historial.
Postcondición si fallo	Si hubiese un fallo, se mostrará un mensaje de error indicando los motivos del error y volverá a mostrar el historial
Actores	Cliente

Secuencia normal	**Paso**	**Acción**
	1	El cliente selecciona la opción historial.
	2	Se mostrará una lista con los servicios solicitados, notificando en color rojo los servicios anulados, en color verde los servicios solicitados ya tramitados y en color blanco aquellos servicios solicitados que aún están pendientes.

	3	Si se presiona sobre un servicio solicitado, nos permite acceder a la opción de Cancelar el servicio.
	3	Si se confirma la cancelación, se comprueba el estado del servicio solicitado. Si no se ha podido cancelar, ir a E1. Si se ha podido cancelar, mostrar mensaje de "Servicio Cancelado" y seguir en el historial. En caso de que se confirme la cancelación, el sistema envía un correo electrónico al usuario confirmando la cancelación.
	4	Si el cliente cancela la operación (botón cancelar), volver a mostrar el historial
Secuencias alternativas	**Paso**	**Acción**
	E1	Se mostrará un mensaje informativo indicando que hubo un problema y volverá a mostrar el historial

3.2.12. Información estática

Muestra información de lugares de interés para el cliente cercanos al hotel.

CASO DE USO	Información estática
Objetivo en contexto	Permite al usuario informarse sobre puntos de interés cerca del hotel, ya sean bares, museos… Además, también podrá mostrar los horarios de autobuses metro y tren cerca del hotel (si hubiera)
Entradas	-
Precondiciones	El cliente debe estar logueado en la aplicación.
Salidas	-
Postcondición si éxito	Si el servicio se ha solicitado correctamente, se mostrará un mensaje indicando que la operación ha sido un éxito y volverá a la pantalla de inicio.
Postcondición si fallo	Si hubiese un fallo, se mostrará un mensaje de error indicando los motivos del error y volverá a la pantalla de inicio
Actores	Cliente

Secuencia normal	**Paso**	**Acción**

	1	El cliente navegará por distintas pestañas donde vendrá información estática sobre puntos de interés. Si hubiese un error en el sistema, ir a E1.
Secuencias alternativas	**Paso**	**Acción**
	E1	Al existir un error saldrá un mensaje de error poniendo "Hubo un error al mostrar la información" e irá posteriormente al menú de la aplicación

3.2.13. Arrancar el servidor

Muestra información de cómo arrancar el servidor

CASO DE USO	Arrancar el servidor
Objetivo en contexto	Permite al responsable de la aplicación poder arrancar el servidor si éste no se encuentra disponible.
Entradas	
Precondiciones	El ordenador donde está alojada la aplicación debe estar encendido. Los servicios de Tomcat y MySql deben estar configurados. Los puertos usados para Tomcat deben estar abiertos y mapeados en el router o firewall.
Salidas	
Postcondición si éxito	Si los servicios se han podido levantar correctamente, la aplicación debería funcionar.
Actores	Responsable de mantenimiento de la aplicación.

Secuencia normal	**Paso**	**Acción**
	1	El responsable de mantenimiento de la aplicación deberá iniciar los servicios de Tomcat y MySql, si estos no estuviesen configurados como servicios arrancables junto con el sistema operativo del servidor. Si alguno de los servicios no se levantasen correctamente, ir a E1.
Secuencias alternativas	**Paso**	**Acción**

	E1	Tendremos que investigar qué puede estar interfiriendo en la ejecución de los servicios.

3.3. Usabilidad

El objetivo en todo momento ha sido diseñar una aplicación fácil de usar y eficiente para el usuario. Tan pronto como el cliente hace el checkin en el hotel, éste recibe un documento indicando un número de usuario y una contraseña asociados a su reserva, evitando así el incómodo proceso de rellenar formularios para registrarse, un proceso que ya nos aburre tanto que muchas veces nos lleva a rechazar el probar una aplicación. En este mismo documento, el cliente también recibirá un código QR con el que poder descargar la aplicación para poder instalarla.

HappH está compuesta por una aplicación móvil y una aplicación web. Ambas aplicaciones comunican con un mismo servidor, por lo que las acciones que se realicen en una aplicación podrán ser consultadas en la otra.

3.3.1 Usabilidad de la aplicación móvil

La aplicación ha sido diseñada de tal forma que el cliente tenga que hacer 3 o 4 movimientos de pestañas como máximo para poder disponer del servicio que desee, ofreciendo así al usuario una navegación de forma rápida, sencilla y sin necesidad de que el usuario tenga grandes conocimientos de su uso o requiera un largo proceso de aprendizaje.

Para facilitar esta navegación rápida, la aplicación está estructurada con una cabecera y un menú que te lleva a los diferentes servicios, buscando así la limpieza y claridad de la aplicación.

Mediante el uso de imágenes aclaratorias y descripciones, se han especificado los diferentes servicios que ofrece la aplicación, convirtiéndola así en una aplicación muy intuitiva y fácil de utilizar.

En la aplicación todos los elementos que se pulsan con los dedos son de un tamaño lo suficientemente grande para que sea fácil y cómodo tocarlos, superando el mínimo recomendado por Google para un dispositivo Android, fijado en 48dp. Además HappH aprovecha toda la amplitud de la pantalla, habiendo sido probada en móviles de diferentes dimensiones.

Por último, para poder utilizar la aplicación es necesaria una conexión de datos, que no tiene por qué ser una conexión wiki, ya que como veremos más adelante los mensajes de comunicación entre aplicación y servidor son breves y concisos, ofreciendo así una gran movilidad al usuario.

Interfaces de usuario

La interfaz comenzará con un formulario de autenticación desde el que se podrá hacer login en la aplicación. A partir de ahí obtendremos una ventana principal donde se encuentran las distintas funcionalidades que se le ofrecen al usuario.

El usuario tiene poca experiencia, por lo que debe de tratarse de una interfaz no muy sobrecargada y con iconos que pueda hacer que sea atractivo para el usuario a la hora de utilizar la aplicación.

Es necesario en este tipo de aplicaciones, que el usuario pueda pedir servicios del hotel en pocos pasos, sin mucha complejidad, y que en cada paso de ventana que esté haciendo el usuario en la aplicación sepa perfectamente en que paso está de la petición. Además, la interfaz de usuario está diseñada en forma de plantillas, de tal forma que sea mucho más sencillo hacer luego posteriores modificaciones para poder personalizar la aplicación de una forma sencilla, además de poder colocar los colores corporativos del hotel en la aplicación.

Para la plantilla base, hemos escogido los colores negro y blanco, ya que en nuestra opinión, hacen de la aplicación algo elegante, y al estar las letras en blanco, remarcan mucho mejor las opciones a las que puede acceder el cliente en HappH. Pero como hemos dicho anteriormente, estos colores son configurables.

3.3.2 Usabilidad de la aplicación web

Teniendo en cuenta que la aplicación móvil está programada para dispositivos Android y que podría suceder que una persona no quisiera descargarse la aplicación, HappH incluye también una aplicación web donde el cliente pueda realizar exactamente las mismas operaciones que a través de la aplicación Android. Esta aplicación web también incluye funcionalidad para que los empleados del hotel puedan gestionar y consultar los servicios, y para que el responsable de gestionar la aplicación pueda configurarla como desee.

Tanto cliente como empleados o administrador disponen de los mismos controles de navegación en todos los módulos que se despliegan en la aplicación. Entre estos controles destacan los siguientes:

- Filtros: Hemos añadido unos formularios con los que poder filtrar los resultados obtenidos en cada uno de los módulos. El usuario de la aplicación podrá por ejemplo filtrar los servicios que hayan sido solicitados en una fecha determinada, filtrar por un tipo de servicio en concreto, por el estado de un servicio, etcétera.

- Límite de resultados: También se limitan los resultados que se muestran en una página, ofreciendo controles para acceder a los siguientes resultados, a los anteriores, a los primeros o a los últimos. La idea de esta limitación es la de no sobrecargar la página con todos los resultados disponibles.

- Orden: Al igual que los filtros, los resultados se ordenan por defecto usando el criterio más apropiado dependiendo del módulo, pero el usuario puede cambiar ese orden colocando por ejemplo primero los resultados con una fecha de solicitud de servicio anterior, ordenarlos por orden alfabético, etcétera.

- Formularios: Los formularios de solicitud de servicios han sido diseñados de forma dinámica, dependiendo del servicio que se quiera solicitar, el formulario se creará con unos campos y controles u otros.

- Mensajes: Toda acción de confirmación conlleva un mensaje informativo. En la aplicación encontramos 3 tipos de mensajes:

    - Mensajes de confirmación, presentados con un icono de un 'check' verde y con un mensaje sobre un fondo verde. Por ejemplo, al terminar la solicitud de un nuevo servicio.

    - Mensajes de eliminación, presentados con un icono de una X roja y con un mensaje sobre fondo rojo. Por ejemplo, cuando un plato del menú no se puede eliminar debido a que hay pedidos solicitados asociados a ese plato.

    - Mensajes de información, presentados con un icono de una exclamación azul, con un mensaje sobre fondo azul. Por ejemplo, para confirmar realmente que un cliente desea anular un servicio solicitado.

También cabe destacar que la aplicación web ha superado los test de accesibilidad WCAG 2.0 (Nivel AA), incluyendo tanto la validación HTML como la validación CSS.

3.4. Fiabilidad

Uno de los elementos fundamentales a la hora de hablar de la fiabilidad de la aplicación es que está desarrollada en la versión 2.3.3 de Android, permitiendo así que la aplicación funcione en versiones posteriores. Todas las pruebas que hemos realizado a lo largo del proyecto se han hecho con dispositivos con versión 2.3.3 y también con dispositivos con versión 4.2.2.

Respecto a la comunicación, la aplicación Android implementa un método que, tras enviar una petición al servidor, lanza un temporizador durante el cual espera respuesta por parte del servidor. Si éste no ha contestado al término de ese temporizador, la aplicación avisa al cliente de que no hay conexión en ese momento, para que vuelva a intentarlo más tarde.

En cuanto a la programación, una de las cosas que más nos ha ayudado a la hora de gestionar el control de errores y el sistema de pruebas ha sido un sistema de log que interpreta las comunicaciones entre el servidor y la aplicación. Este mismo log estaría accesible para el responsable de la aplicación, para la gestión de incidencias.

También como veremos más adelante en el apartado de rendimiento, hemos sometido tanto a la aplicación como al servidor a varios test de estrés, lanzando peticiones simultáneas tanto de lectura de datos como de escritura, superando con éxito todos los test que hemos planteado.

Hablando de la consistencia de los datos, hemos tenido muy presente las relaciones entre los mismos, controlando en todo momento la integridad referencial a la hora de realizar operaciones que impliquen modificación o eliminación de datos. Así, el sistema no permitirá por ejemplo eliminar una categoría de coches si hay coches asociados a dicha categoría. Tampoco se permite la anulación de servicios una vez hayan sido atendidos, o hayan sido anulados con anterioridad.

Uno de los puntos pendientes del proyecto es la consistencia de la información que es gestionada por HappH. Al término del proyecto no tenemos un sistema que realice copias de seguridad de la base de datos. Una posible solución sería que el sistema, tras el primer acceso de un usuario pasadas las 00:00h, realice un volcado de la base de datos a un fichero, almacenando por ejemplo 7 copias, una para cada día de la semana.

3.5. Rendimiento y compatibilidad

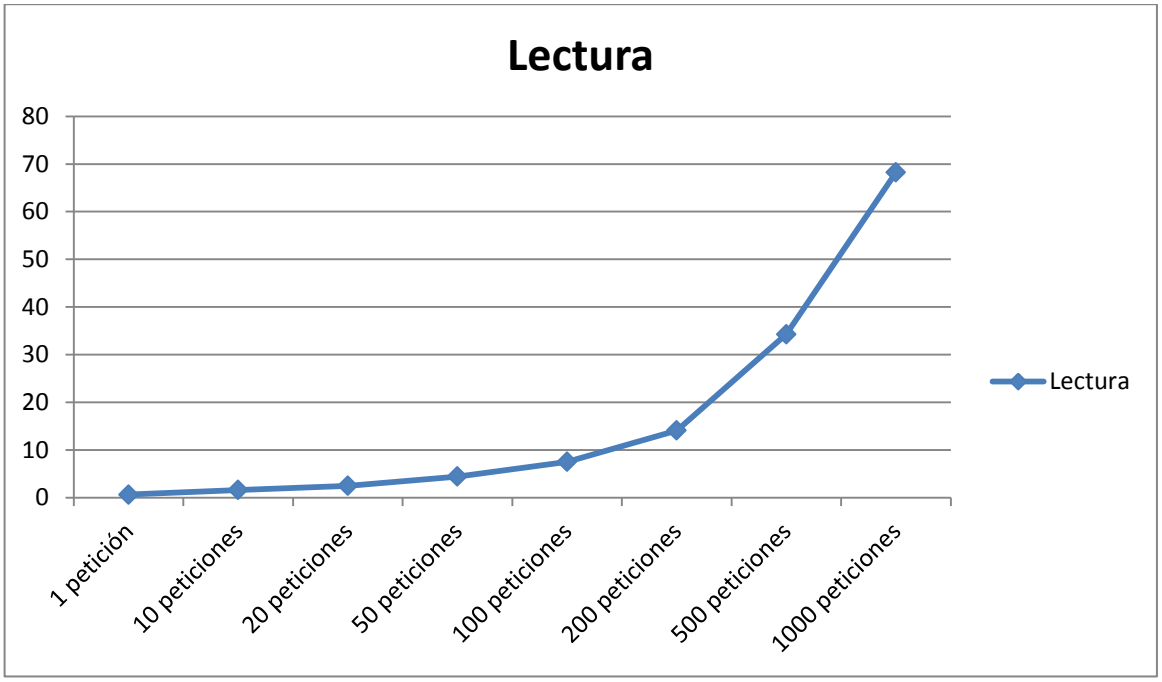
3.5.1 Rendimiento

Al término de este proyecto, la aplicación se encuentra alojada en un ordenador que hace las funciones de servidor. Éste ordenador ha estado encendido las 24 horas del día, y a pesar de contar con una red doméstica de internet, ha sido capaz de atender todas las peticiones de servicios que hemos realizado para probar la aplicación, sometiéndolo a varios test de estrés con los siguientes resultados:

- Test de lectura: Hemos realizado un test de peticiones de login realizando 1 petición, 10, 20, 50, 100, 200, 500 y 1000 peticiones simultáneas:

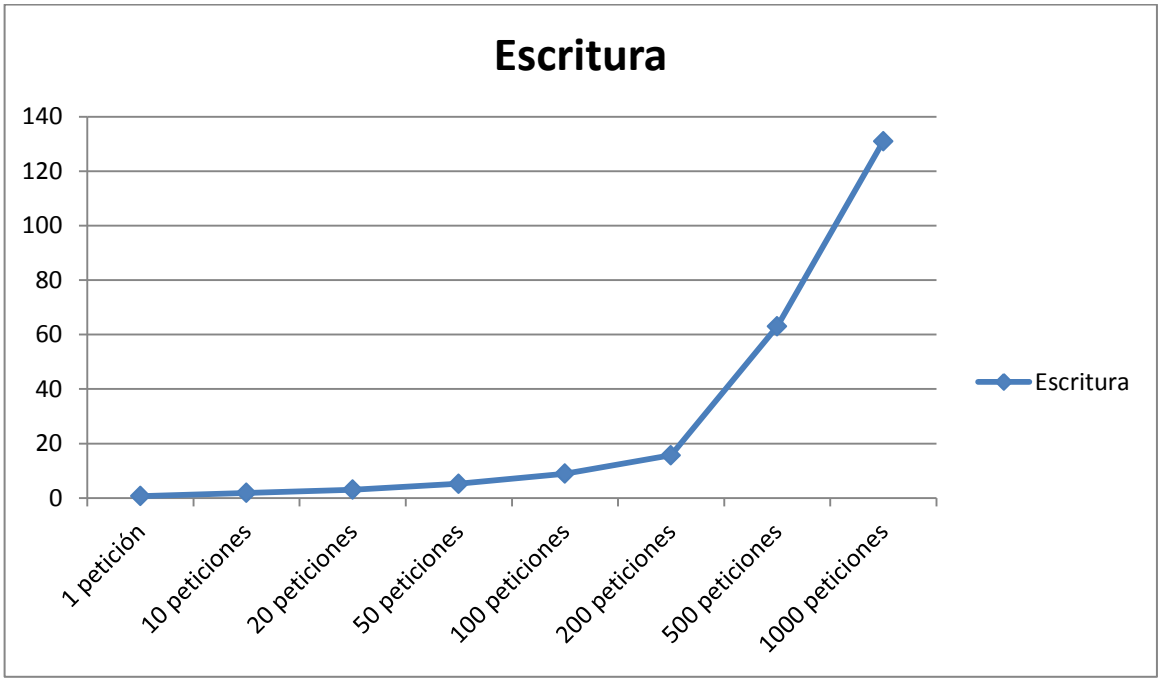
Número de peticiones	Tiempo de ejecución (en segundos)	Tiempo medio por petición (en segundos)
1	0,653	0,653

10	1,625	0,1625
20	2,485	0,12425
50	4,423	0,08846
100	7,519	0,07519
200	14,114	0,07057
500	34,294	0,06859
1000	68,289	0,06829



- Test de escritura: Hemos realizado un test de solicitud de servicios realizando 1 petición, 10, 20, 50, 100, 200, 500 y 1000 peticiones simultáneas:

Número de peticiones	**Tiempo de ejecución (en segundos)**	**Tiempo medio por petición (en segundos)**
1	0,688	0,688
10	1,884	0,1884
20	3,056	0,1528
50	5,224	0,1045
100	8,951	0,0895
200	15,723	0,0786
500	63,058	0,1261
1000	130,988	0,1310



Analizando los resultados observamos la estabilidad del servidor, capaz de albergar hasta 1000 peticiones de servicios simultáneas. Observamos también como para las peticiones de lectura, el tiempo promedio para ejecutar las peticiones de login disminuye a medida que hemos ido aumentando el número de peticiones simultáneas. Lo mismo sucede con las peticiones de grabación de servicios, hasta llegar a las 500 peticiones, donde el tiempo promedio vuelve a aumentar.

3.5.2. Compatibilidad

En cuanto a la compatibilidad de la aplicación, está desarrollada para el sistema Android 2.3.3, ya que es la versión que cubre la mayor parte de las aplicaciones que se ofrecen en el mercado de Google. Sin embargo, esta aplicación se ha probado también en versiones posteriores de Android con resultados significativos (probado en la versión 4.1.2 en móvil y la versión 3.0 para Tablet). Los resultados en Tablet no fueron visualmente muy buenos, ya que esta aplicación está desarrollada para plataforma móvil, sin embargo, pudimos ver que en Tablet también se podían hacer las peticiones de forma normal. Además, elegimos en su momento desarrollar la aplicación con Android y no utilizando HTML5 ya que uno de los puntos que se querían alcanzar era el de aprender a desarrollar con las interfaces que proporciona Android.

Como hemos comentado con anterioridad, para aumentar la compatibilidad de nuestro proyecto hemos desarrollado una aplicación web accesible por 3 tipos diferentes de personas:

- Clientes, que podrán realizar las mismas gestiones que con la aplicación Android, es decir, solicitar servicios, ver en qué estado están sus solicitudes y anular un servicio.

- Empleados, quienes podrán administrar y gestionar los servicios solicitados y hacer tanto el checkin como el checkout de los clientes.

- Administrador del sistema, quien podrá configurar que empleados tienen acceso a la aplicación, que servicios se ofrecen y a qué precios, configurar por ejemplo los coches ofertados en el servicio de alquiler, los platos que se pueden solicitar en el servicio de comidas, etcétera.

En cuanto a los restantes sistemas operativos del mercado (como Windows Phone o IOS), cabe decir que siempre será compatible con el servidor ya que la parte más crítica es la comunicación entre el servidor y la aplicación HappH. Mientras se disponga de un mecanismo para poder crear y parsear mensajes en formato XML, no habrá ningún impedimento para poder desarrollar la aplicación en diferentes plataformas.

4. ARQUITECTURA DEL SISTEMA

En este punto se presentará un resumen de la arquitectura software utilizada en el sistema HappH, analizando los diferentes aspectos del mismo y las decisiones arquitectónicas tomadas.

4.1 Objetivos del diseño

El objetivo del proyecto ha sido la creación de un sistema de gestión hotelera que englobe principalmente las peticiones de servicios por parte de los clientes y el control tanto de los clientes como de los servicios ofertados a cargo del personal. Las peticiones se realizarán a través de una aplicación Android y de una interfaz web y la gestión será a través de ésta última.

Se ha buscado sustituir el modo tradicional utilizado, bien sea mediante llamada o comunicación en persona con la recepción del hotel, por la comunicación mediante dispositivo móvil.

Este medio obliga a la notificación al usuario del éxito en las peticiones de servicios, así como un control sobre las ya realizadas y la posibilidad de cancelación, en los términos acordados por el hotel, de las que considere. También es necesario un control de acceso para garantizar la seguridad del sistema, al igual que considerar las eventuales retiradas de los servicios del catálogo de los ofrecidos, bien sean temporales o permanentes.

Un objetivo importante del proyecto ha sido la búsqueda de una fácil extensibilidad a nuevos servicios para ser capaz de adaptarse modularmente a las necesidades de cada hotel concreto.

4.2 Servidor web

Teniendo en cuenta los requisitos iniciales, la arquitectura Java EE es la que mejor se adapta para el desarrollo de este proyecto, brindando una arquitectura multicapa y distribuida.

Las capas lógicas de las que se compone la aplicación son las siguientes:

- Cliente: se conecta a Internet y accede a la aplicación desde un navegador.
- Vista: representa la interfaz visual con la que interactuará el usuario.
- Modelo: contiene las reglas de negocio del sistema.
- Persistencia: almacena y recupera la información de un motor de base de datos.

Entre las principales ventajas de utilizar esta arquitectura multicapa encontramos:

- Mantenimiento: una modificación en una capa no implica la modificación del resto de capas de la aplicación.

- Reusabilidad: Los datos y el modelo sólo se definen una vez, permitiendo a otras aplicaciones utilizarlos sin violar las reglas del sistema.
- Escalabilidad: el sistema es flexible para dividirse físicamente cuando los requerimientos sobre el desempeño de la aplicación cambian.
- Cliente ligero: Los usuarios pueden conectarse a la aplicación sin importar el sistema operativo que utilizan y beneficiarse de mejoras en el sistema sin necesidad de descargar ningún software adicional.

Hay que tener en cuenta que para facilitar el desacoplamiento entre la Vista y el Modelo se utiliza el patrón de diseño Modelo-Vista-Controlador (MVC). Al utilizar MVC, los datos de negocio están separados de la lógica de presentación, y los componentes que procesan los datos no controlan la visualización de los mismos y viceversa.

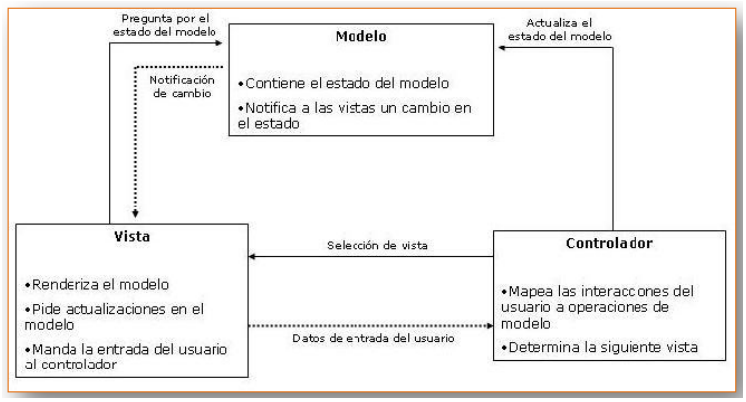


Ilustración 3. Modelo Vista Controlador

Para la implementación física de esta arquitectura se han tomado las siguientes decisiones que afectan a la fase de desarrollo:

- Java: Java RuntimeEnvironmentversion“1.7”.
- Presentación: JavaServerPages (JSP) para generar las páginas web de la aplicación e implementar el controlador de la arquitectura MVC.
- Persistencia: Se utiliza MySql debido a que se trata de un motor de base de datos gratuito y extensamente utilizado e Hibernate. La ventaja de utilizar Hibernate frente a JDBC es que nos permite mapear las relaciones y atributos de nuestras tablas en objetos e independizar el código de nuestra aplicación de un cambio de la base de datos.
- Servidor web: se utiliza como servidor web Apache Tomcat ya que es gratuito y de los más usados. Este servidor implementa las especificaciones de Servlets y JSP que necesitaremos para ofrecer el acceso a la aplicación a través de internet.

4.2.1 Modelo Relacional

A continuación se presenta el modelo relacional utilizado en la aplicación para gestionar los datos de empleados, clientes, reservas, servicios ofrecidos por el hotel y servicios solicitados por los clientes.

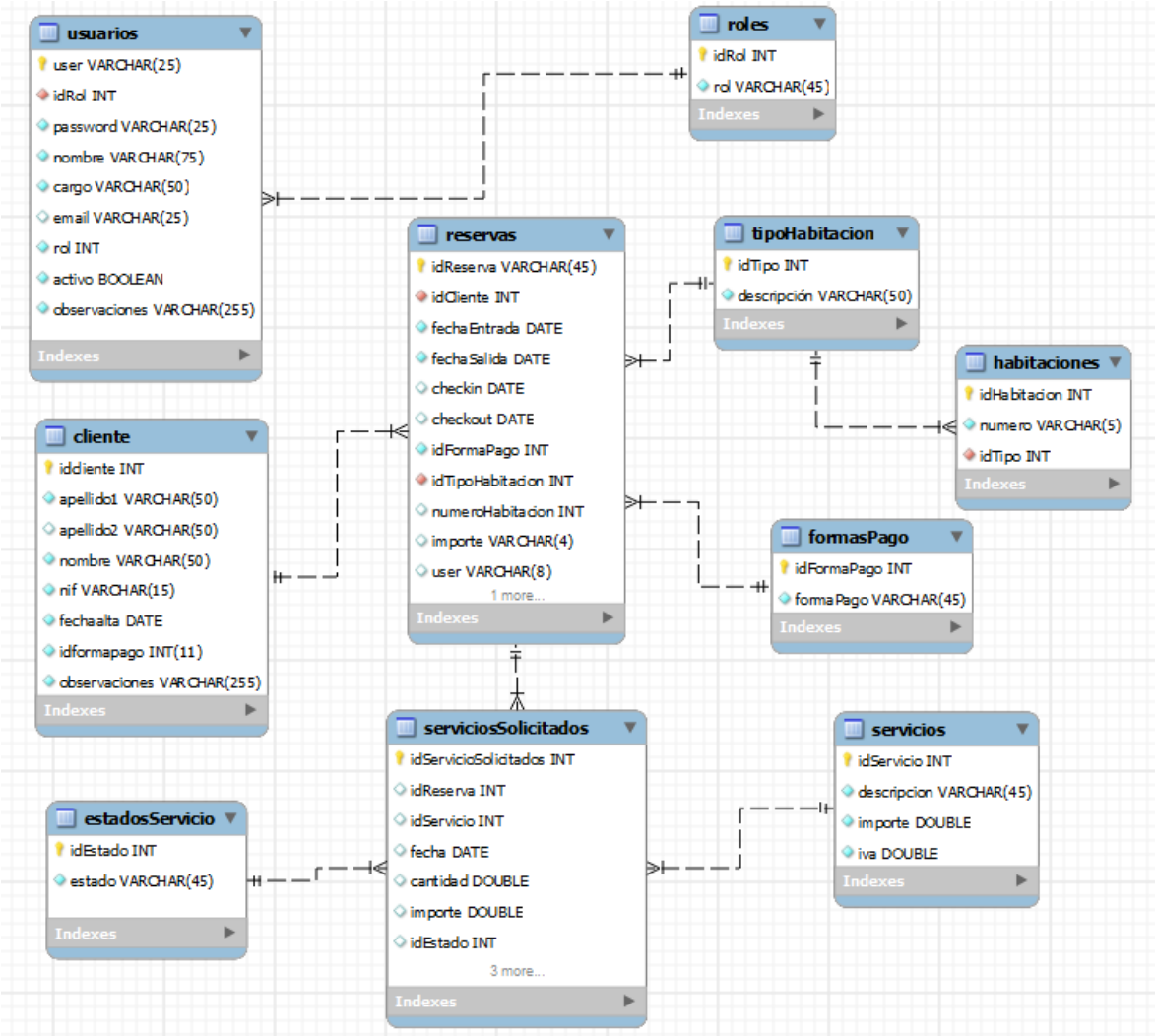


Ilustración 4. Modelo relacional

4.2.2. Diagramas de Clases

Patrón DAO

El patrón DAO es una solución al problema de acoplamiento entre una aplicación orientada a objetos y una base de datos relacional, empleando únicamente la interfaz de programación nativa del gestor de base de datos o algún otro sustituto como JDBC, ODBC, etcétera.

Se utiliza para:

- Abstraer y encapsular los accesos
- Gestionar las conexiones a la fuente de datos
- Obtener los datos almacenados

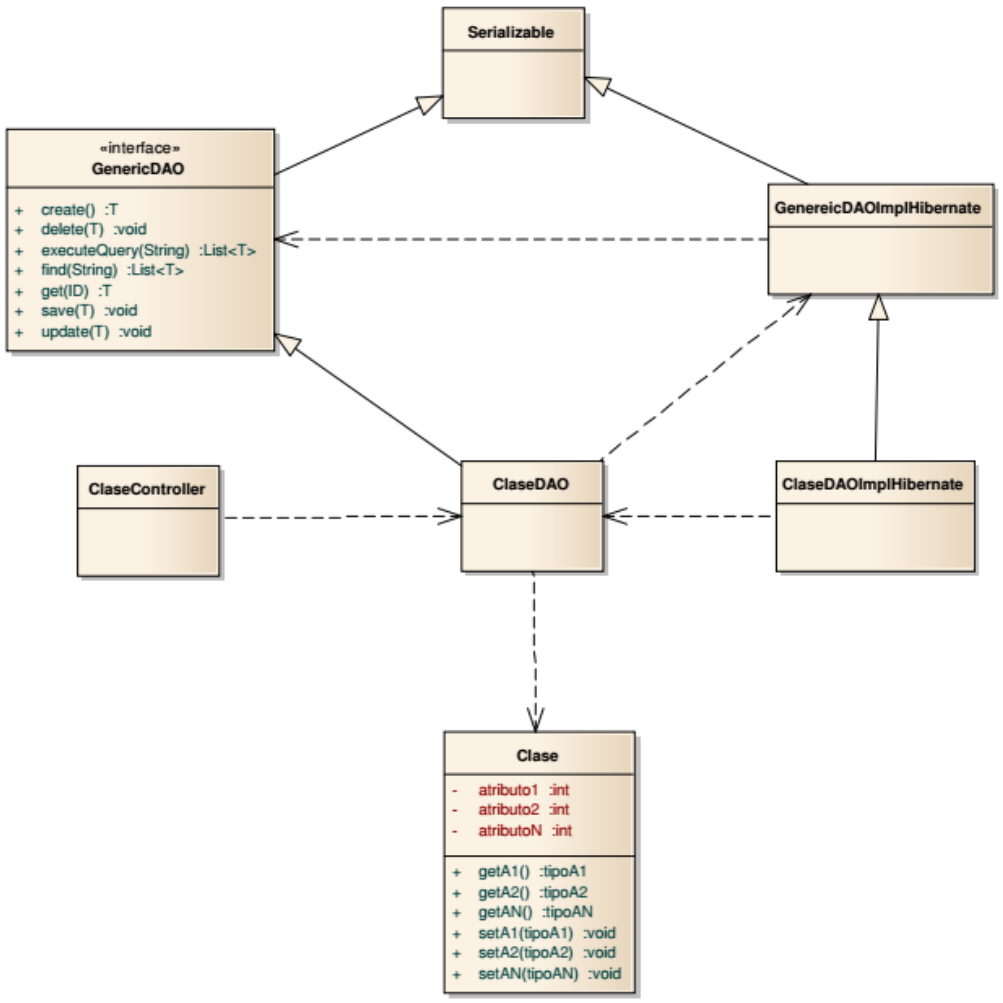


Ilustración 5. Diagrama de clases. Patrón DAO.

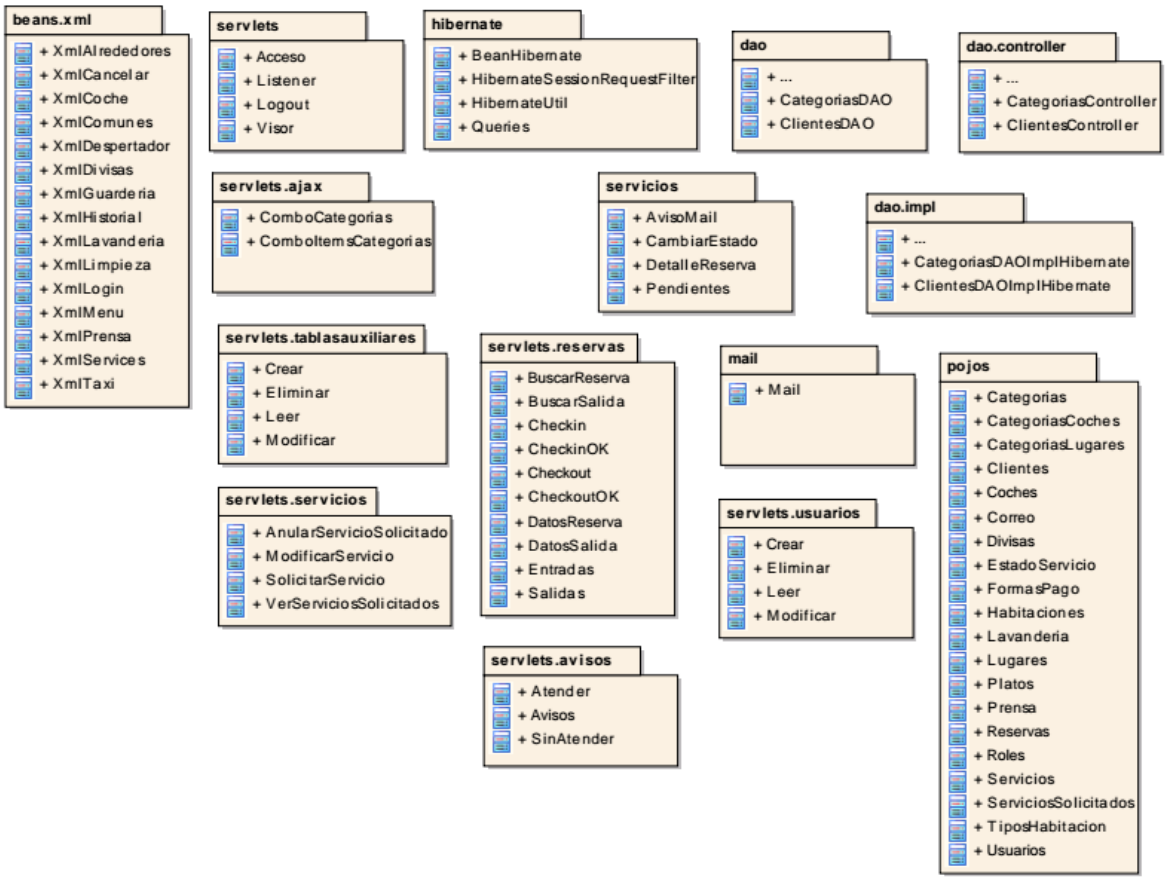


Ilustración 6. Servidor web – Paquete de clases.

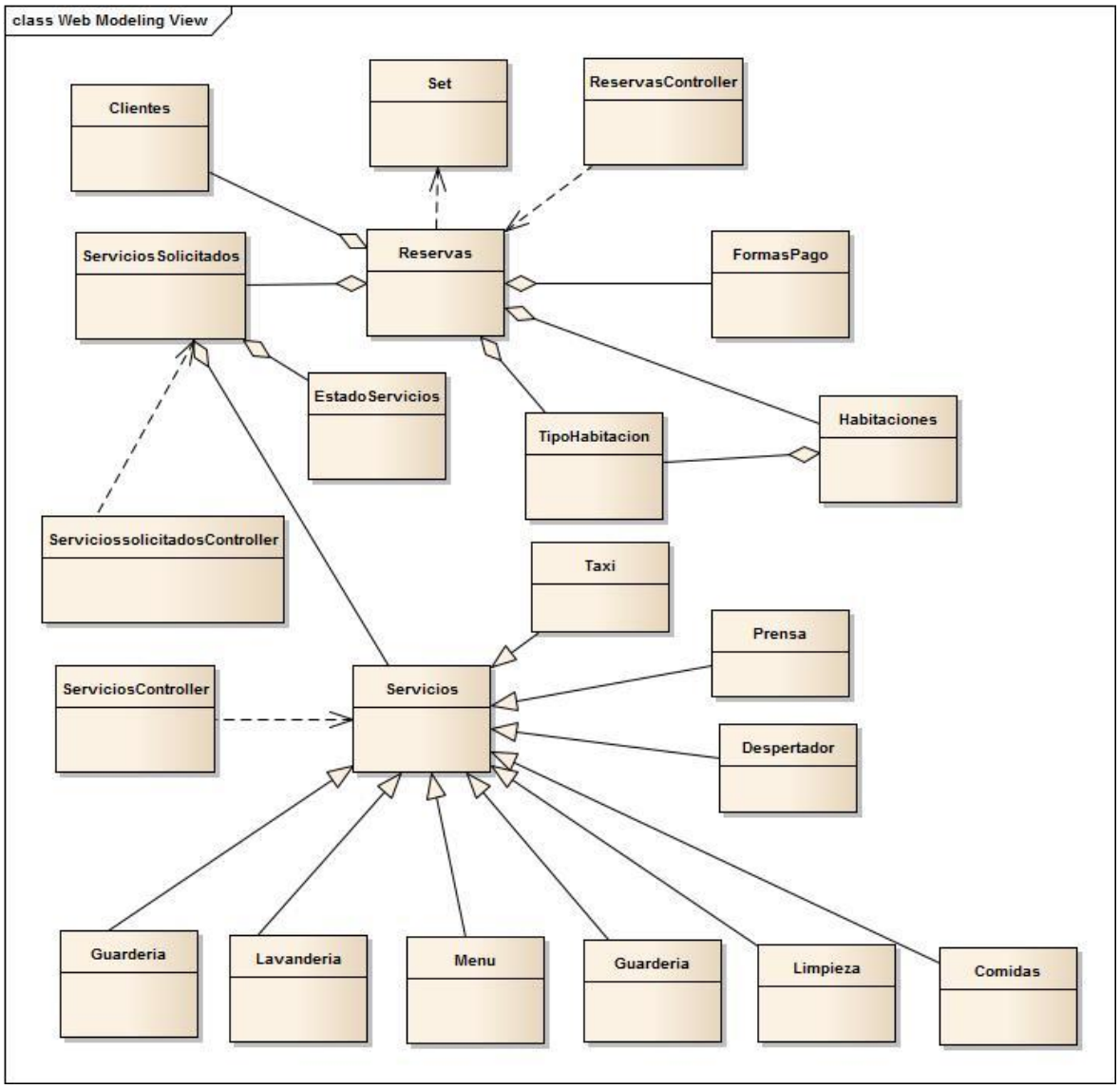


Ilustración 7. Diagrama de clases

4.2.2. Vista de interacción

4.2.2.1. Login

Se muestra la interacción entre la aplicación móvil y el servidor web durante la operación de login en el sistema.

La aplicación Android contacta por http con el servlet Listener encargado de escuchar, tramitar y enviar las respuestas a todas las solicitudes que le llegan. En este caso el Servlet llama a la clase XMLComunes que valida al cliente.

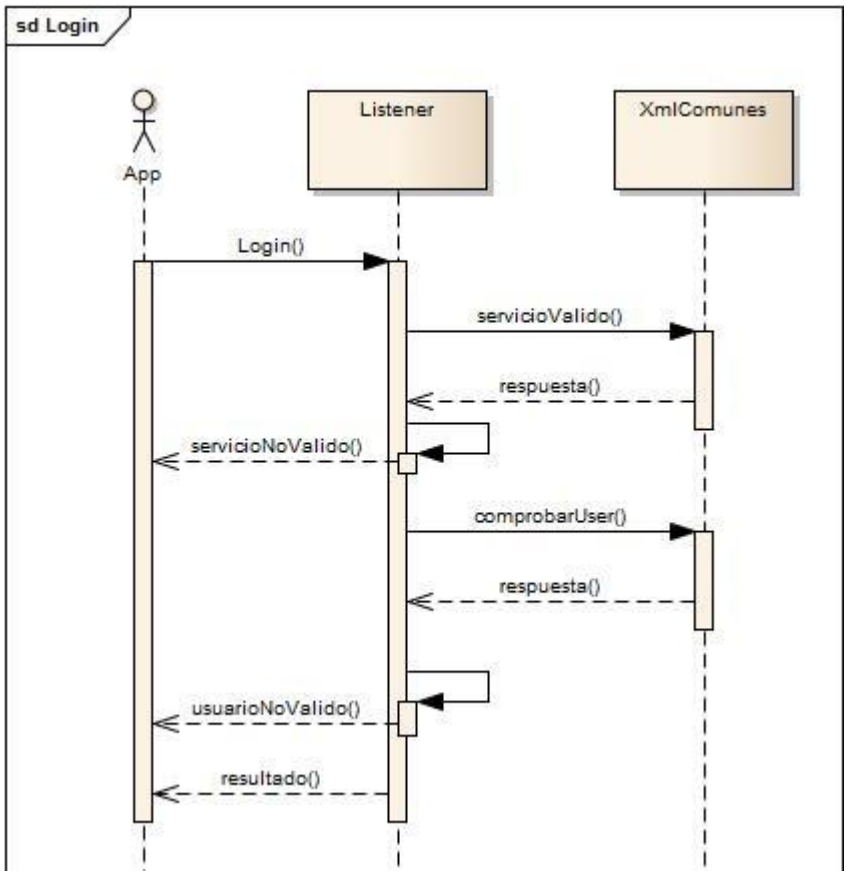


Ilustración 8. Patrón DAO

4.2.2.2. Servicio de taxi

Se muestra la interacción entre la aplicación móvil y el servidor web durante la petición de un taxi. El servlet Listener recibe los datos del cliente y el servicio solicitado. XmlComunes valida los datos, si todo está correcto Listener pasa la solicitud a XmlTaxi el cual llama al controlador de reservas XmlReservas que se encarga de buscar la reserva del cliente, crear un nuevo servicio solicitado, grabar los datos y devolver el control al servlet con el estado de la operación que será devuelto al cliente.

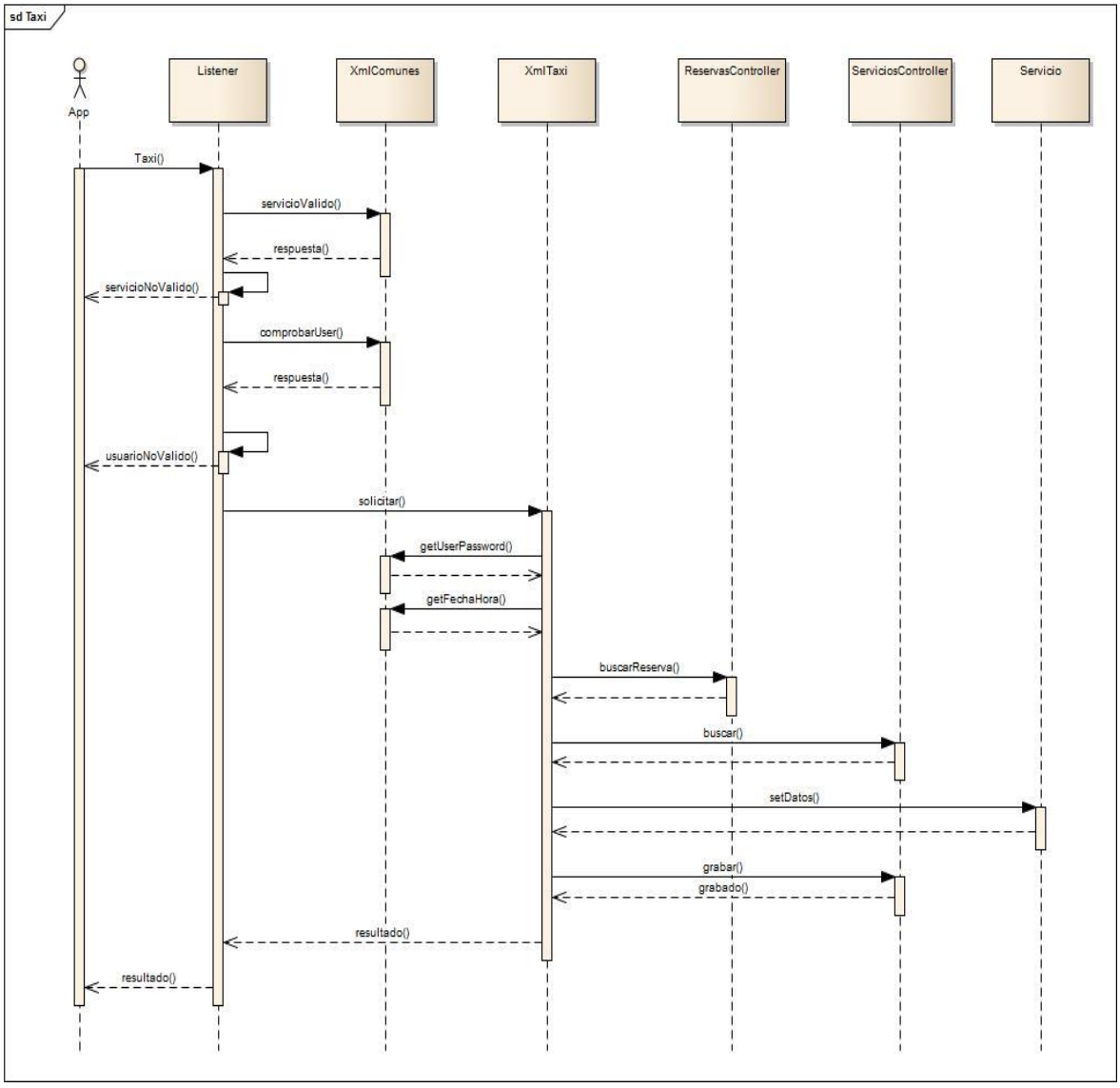


Ilustración 9. Diagrama de secuencia servidor – Servicio de taxi

4.2.2.3. Servicio de limpieza

Se muestra la interacción entre la aplicación móvil y el servidor web durante la petición del servicio de limpieza. El servlet Listener recibe los datos del cliente y el servicio solicitado. XmlComunes los valida, si todo está correcto Listener pasa la solicitud a XmlLimpieza el cual llama al controlador de reservas XmlReservas que se encarga de buscar la reserva del cliente, crear un nuevo servicio solicitado, grabar los datos y devolver el control al servlet con el estado de la operación que será devuelto al cliente.

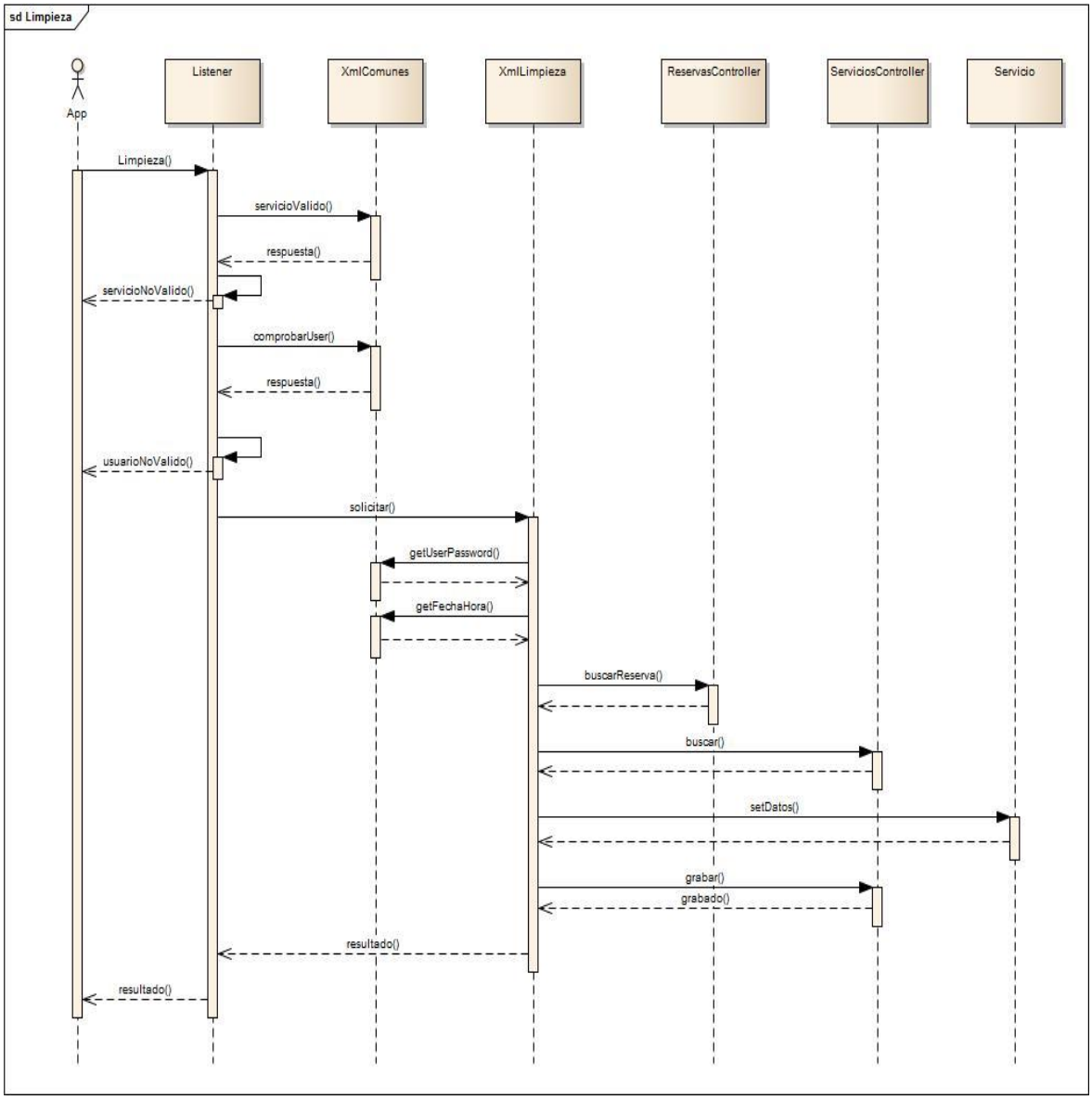


Ilustración 10. Diagrama de secuencia servidor – Servicio de limpieza

4.2.2.4. Servicio de lavandería

Se muestra la interacción entre la aplicación móvil y el servidor web durante la petición del servicio de lavandería. El servlet Listener recibe los datos del cliente y el servicio solicitado. XmlComunesvalida estos datos, si todo está correcto Listener pasa la solicitud a XmlLavanderia el cual llama PrendasController que le suministra la lista de posibles prendas a lavar al servlet y éste al cliente.

En una segunda fase el cliente le indica al servlet Listener qué prendas desea lavar, se vuelven a validar los datos en XmlComunes, si todo es correcto se pasa el control al XmlLavanderia el cual llama al controlador de reservas XmlReservas que se encarga de buscar la reserva del cliente, crear un nuevo servicio solicitado, grabar los datos y devolver el control al servlet con el estado de la operación que será devuelto al cliente.

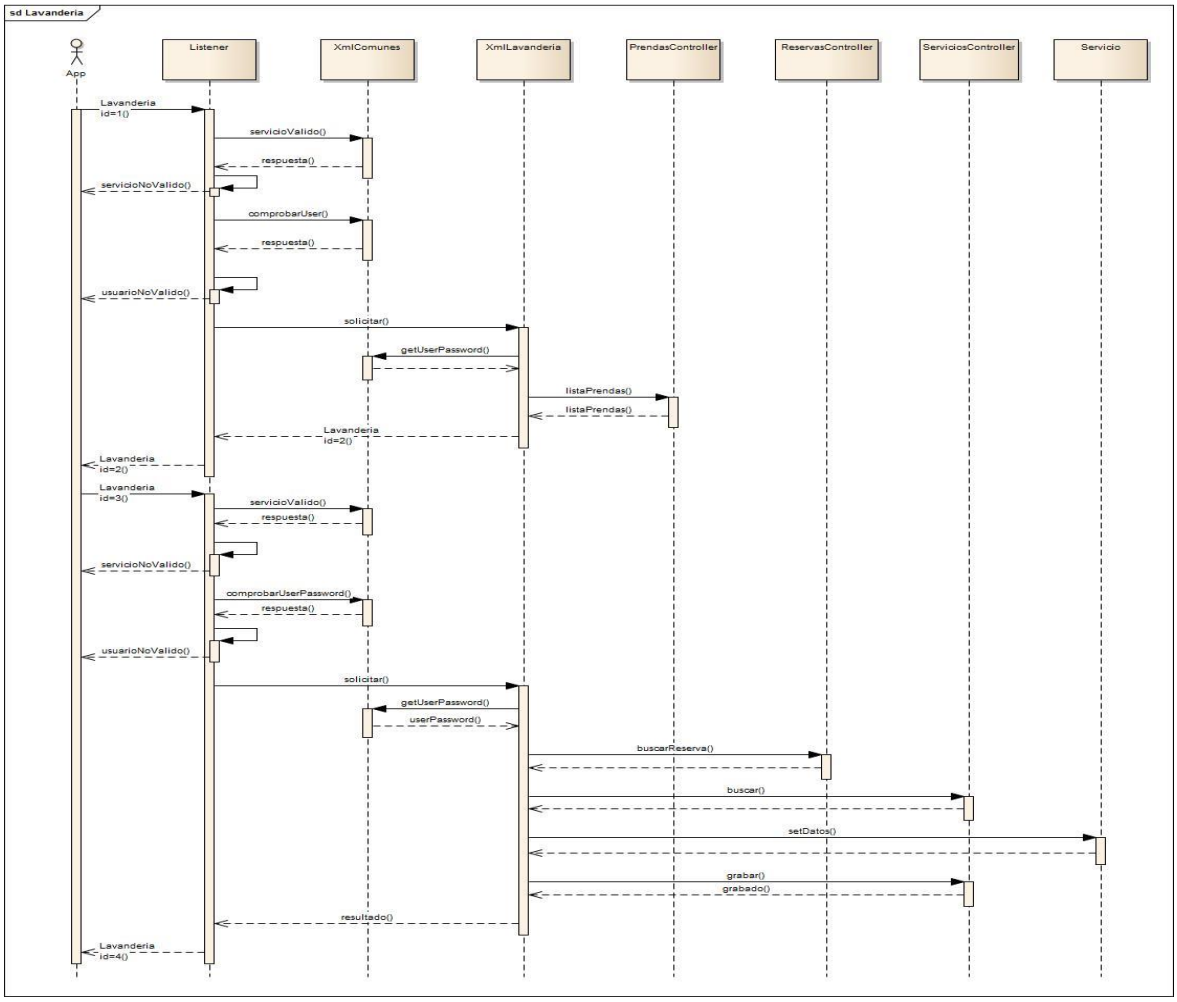


Ilustración 11. Diagrama de secuencia servidor – Servicio de lavandería

4.2.2.5. Servicio despertador

Se muestra la interacción entre la aplicación móvil y el servidor web durante la petición del servicio despertador. El servlet Listener recibe los datos del cliente y el servicio solicitado. XmlComunes los valida, si todo está correcto Listener pasa la solicitud a XmlDespertador el cual llama al controlador de reservas XmlReservas que se encarga de buscar la reserva del cliente, crear un nuevo servicio solicitado, grabar los datos y devolver el control al servlet con el estado de la operación que será devuelto al cliente.

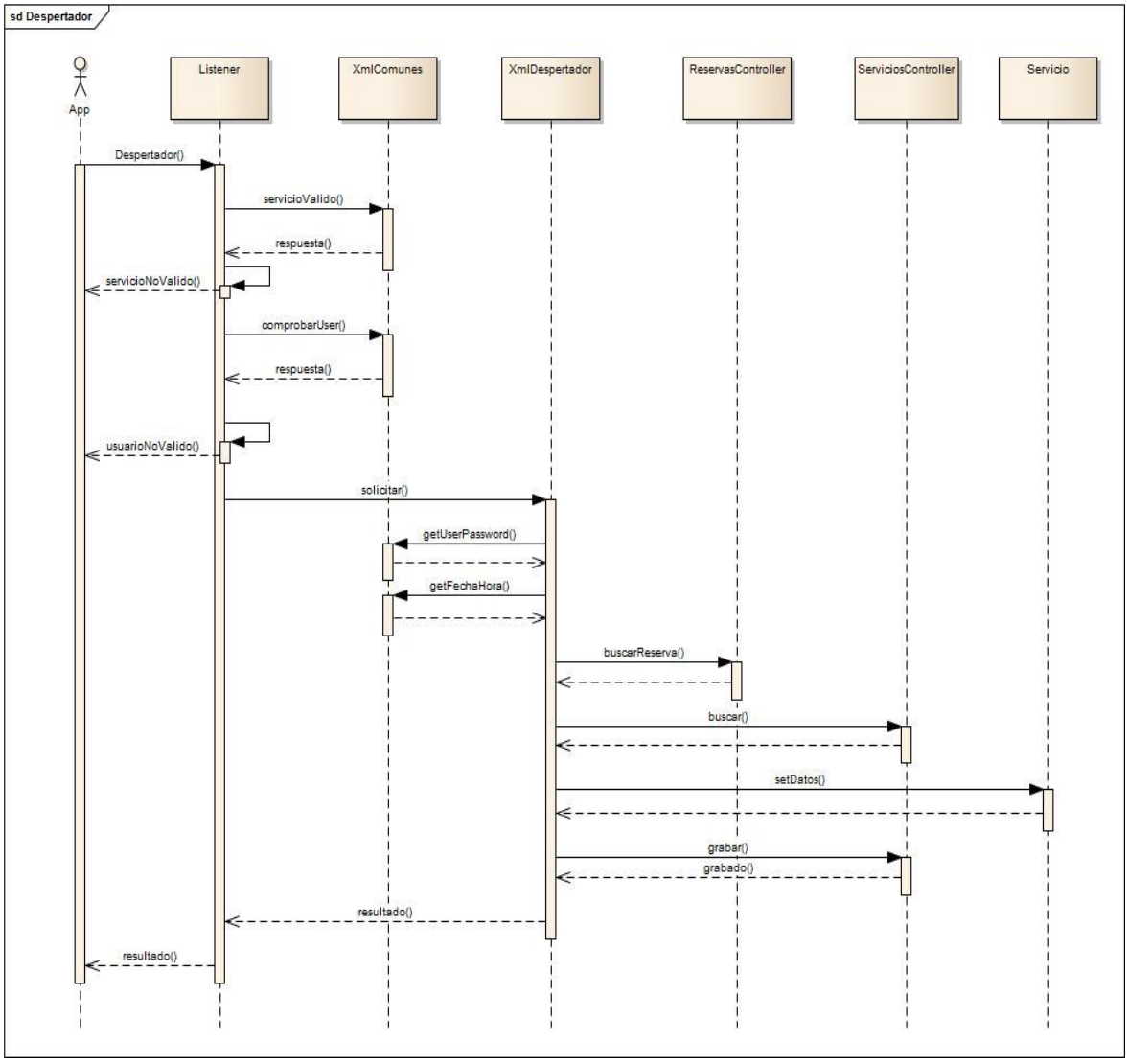


Ilustración 12. Diagrama de secuencia servidor – Servicio despertador

4.2.2.6. Servicio de guardería

Se muestra la interacción entre la aplicación móvil y el servidor web durante la petición del servicio de guardería. El servlet Listener recibe los datos del cliente y el servicio solicitado. XmlComunes los valida, si todo está correcto Listener pasa la solicitud a XmlGuarderia el cual llama al controlador de reservas XmlReservas que se encarga de buscar la reserva del cliente, crear un nuevo servicio solicitado, grabar los datos y devolver el control al servlet con el estado de la operación que será devuelto al cliente.

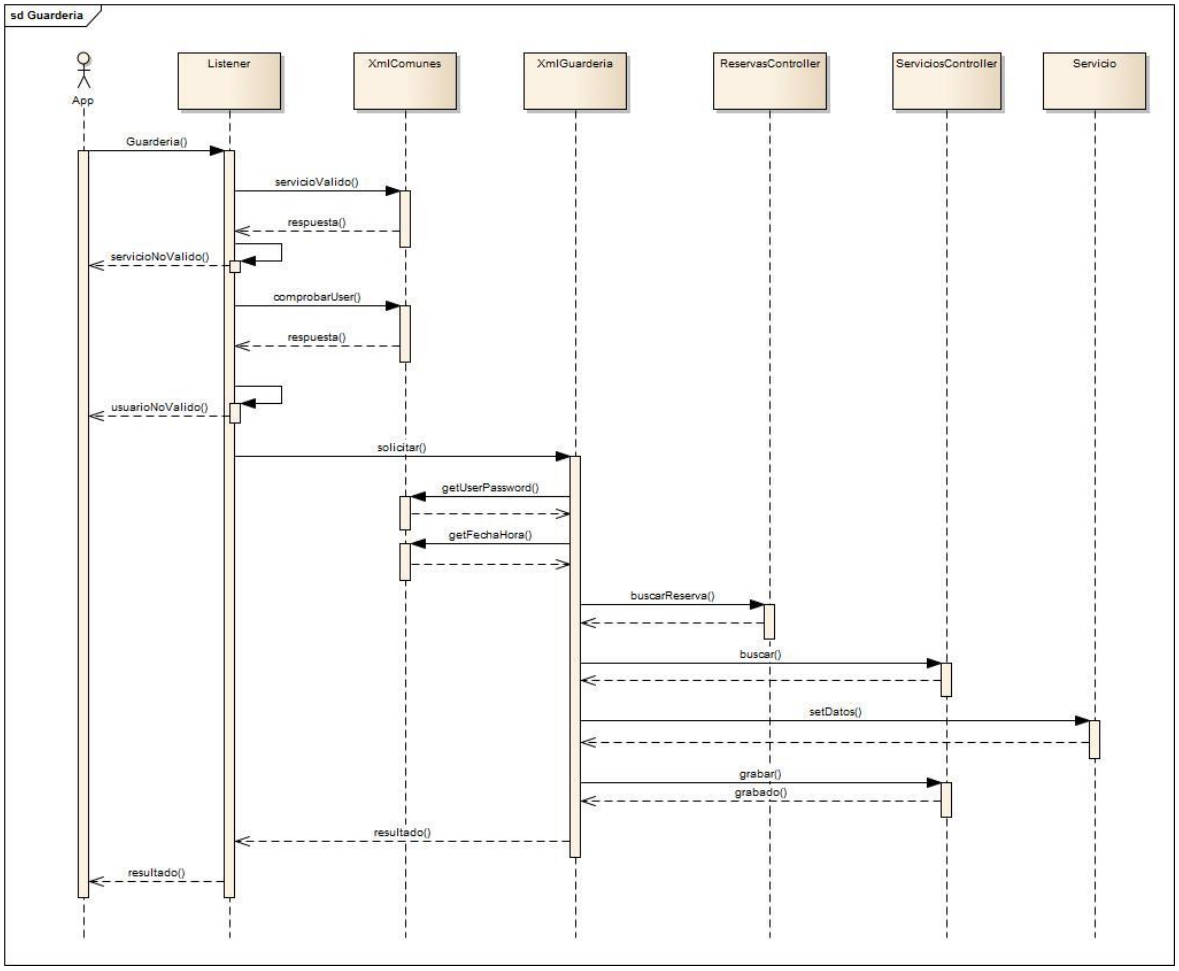


Ilustración 13. Diagrama de secuencia servidor – Servicio de guardería

4.2.2.7. Servicio de comidas

Se muestra la interacción entre la aplicación móvil y el servidor web durante la petición del servicio de comidas. El servlet Listener recibe los datos del cliente y el servicio solicitado. XmlComunes valida estos datos, si todo está correcto Listener pasa la solicitud a XmlMenu el cual llama CategoriasController que le suministra las categorías disponibles de platos al servlet y éste al cliente.

En una segunda fase el cliente le indica al servlet Listener qué categoría de platos desea consultar, se vuelven a validar los datos en XmlComunes, si todo es correcto se pasa el control a XmlMenu, el cual llama al controlador de platos para obtener la lista de platos de la categoría seleccionada y devolverla al cliente.

En la última fase, el cliente le indica al servlet Listener qué plato ha elegido. El servlet vuelve a validar el mensaje usando XmlComunes, si todo es correcto pasa la solicitud a XmlMenu el cual llama al controlador de reservas XmlReservas que se encarga de buscar la reserva del cliente, crear un nuevo servicio solicitado, grabar los datos y devolver el control al servlet con el estado de la operación que será devuelto al cliente.

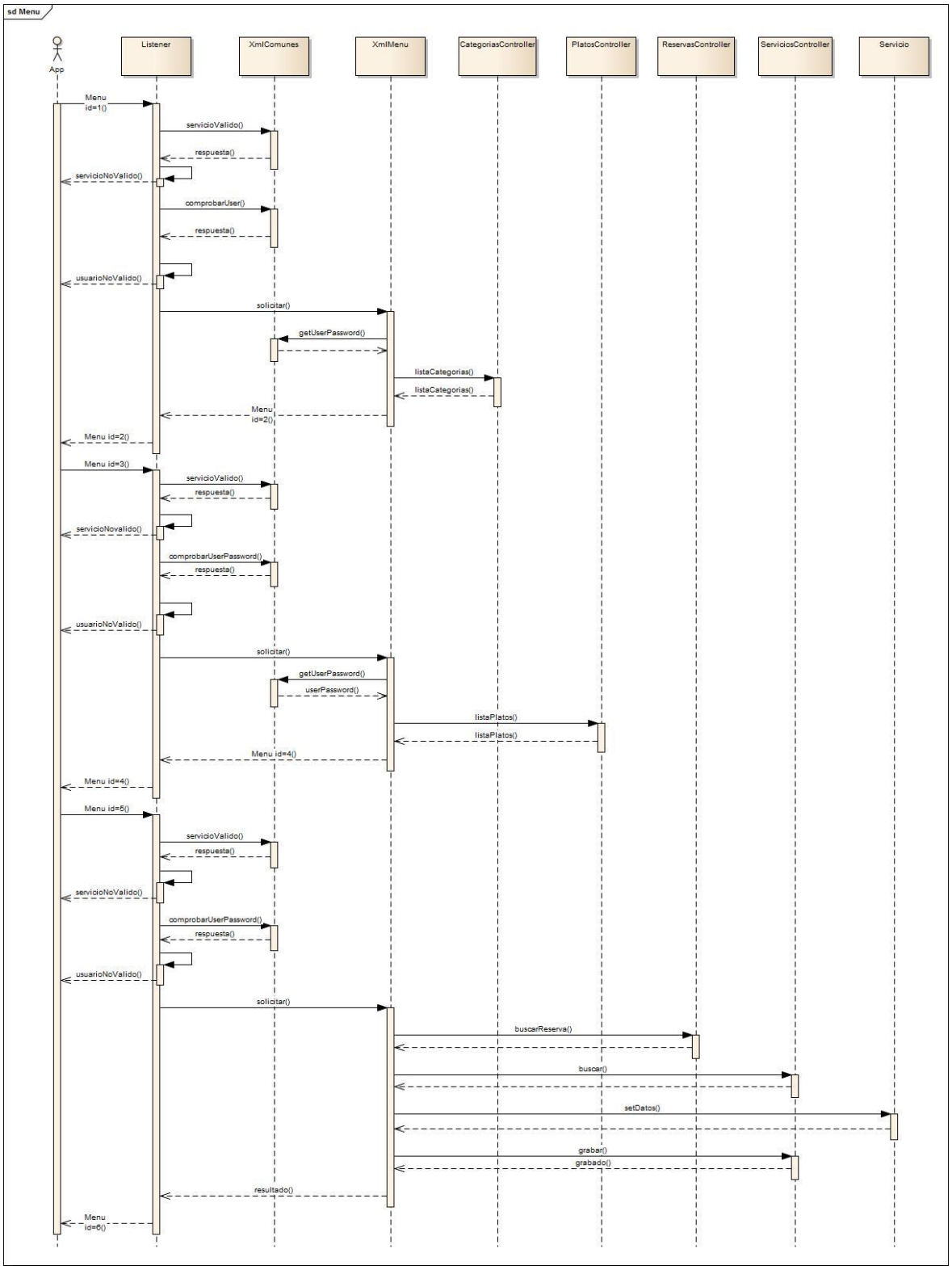


Ilustración 14. Diagrama de secuencia servidor – Servicio de comidas

4.2.2.8. Servicio de prensa

Se muestra la interacción entre la aplicación móvil y el servidor web durante la petición del servicio de prensa. El servlet Listener recibe los datos del cliente y el servicio solicitado. XmlComunesvalida los datos, si todo está correcto Listener pasa la solicitud a XmlPrensa el cual llama PrensaController que le suministra la lista de periódicos y revistas, se las pasa al servlet y éste al cliente.

En una segunda fase el cliente le indica al servlet Listener qué prendas desea lavar, se vuelven a validar los datos en XmlComunes, si todo es correcto se pasa el control al XmlPrensa, el cual llama al controlador de reservas XmlReservas que se encarga de buscar la reserva del cliente, crear un nuevo servicio solicitado, grabar los datos y devolver el control al servlet con el estado de la operación que será devuelto al cliente.

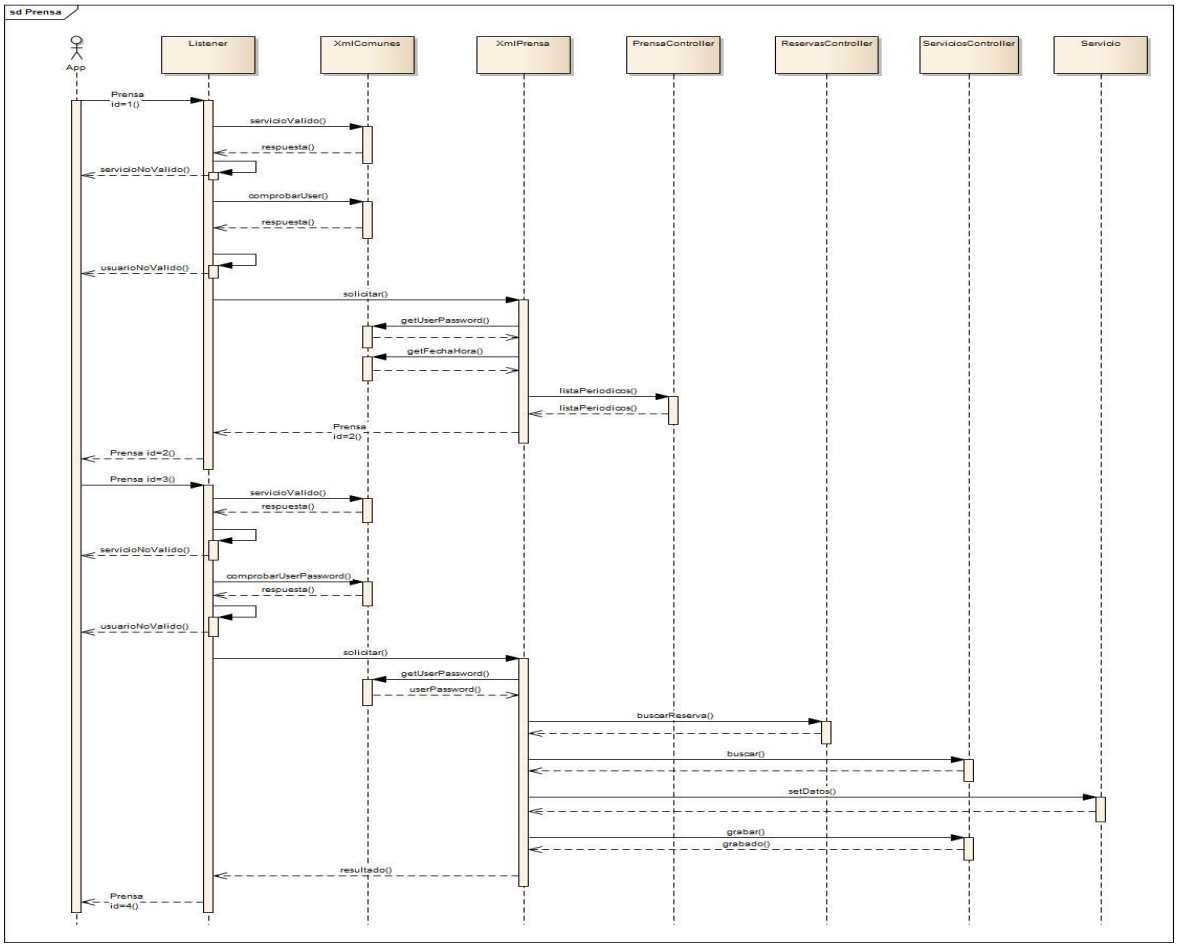


Ilustración 15. Diagrama de secuencia servidor – Servicio de prensa

4.2.2.9. Servicio de cambio de divisas

Se muestra la interacción entre la aplicación móvil y el servidor web durante la petición del servicio de cambio de divisas. El servlet Listener recibe los datos del cliente y el servicio solicitado. XmlComunes valida los, si todo está correcto Listener pasa la solicitud a XmlDivisas el cual llama DivisasController que le suministra la lista de divisas con sus tasas de cambio.

En una segunda fase el cliente le indica al servlet Listener qué prendas desea lavar, se vuelven a validar los datos en XmlComunes, si todo es correcto se pasa el control al XmlDivisas, el cual llama al controlador de reservas XmlReservas que se encarga de buscar la reserva del cliente, crear un nuevo servicio solicitado, grabar los datos y devolver el control al servlet con el estado de la operación que será devuelto al cliente.

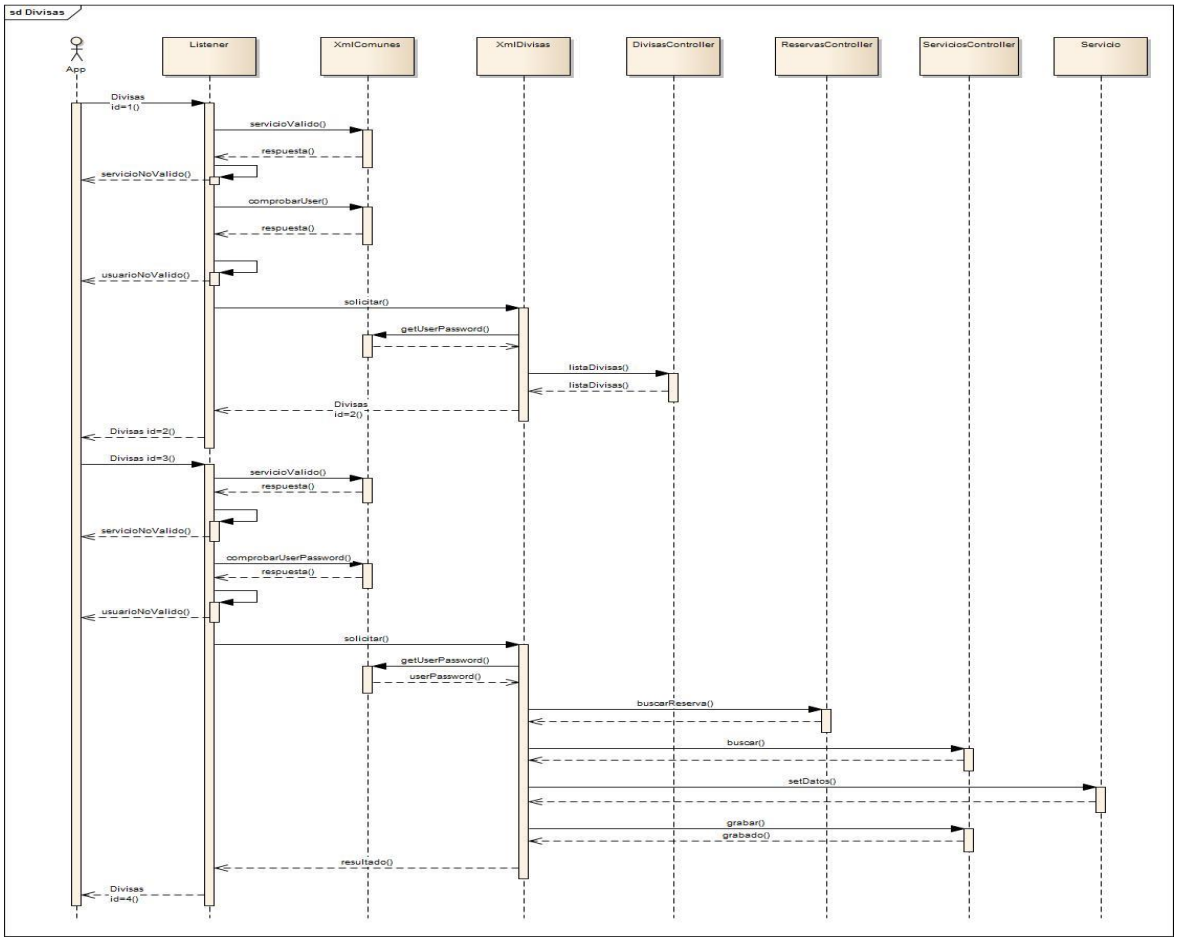


Ilustración 16. Diagrama de secuencia servidor – Servicio de cambio de divisas

4.2.2.10. Servicio de alquiler de coches

Se muestra la interacción entre la aplicación móvil y el servidor web durante la petición del servicio de alquiler de coches. El servlet Listener recibe los datos del cliente y el servicio solicitado. XmlComunes valida los datos, si todo está correcto Listener pasa la solicitud a XmlAlquiler el cual llama CategoriasCochesController que le suministra las categorías disponibles de vehículos al servlet y éste al cliente.

En una segunda fase el cliente le indica al servlet Listener qué categoría de platos desea consultar, se vuelven a validar los datos en XmlComunes, si todo es correcto se pasa el control a XmlAlquiler, el cual llama al controlador de platos CocherController para obtener la lista de vehículos de la categoría seleccionada y devolverla al cliente.

En la última fase, el cliente le indica al servlet Listener qué coche ha elegido. El servlet vuelve a validar el mensaje usando XmlComunes, si todo es correcto pasa la solicitud a XmlAlquiler el cual llama al controlador de reservas XmlReservas que se encarga de buscar la reserva del cliente, crear un nuevo servicio solicitado, grabar los datos y devolver el control al servlet con el estado de la operación que será devuelto al cliente.

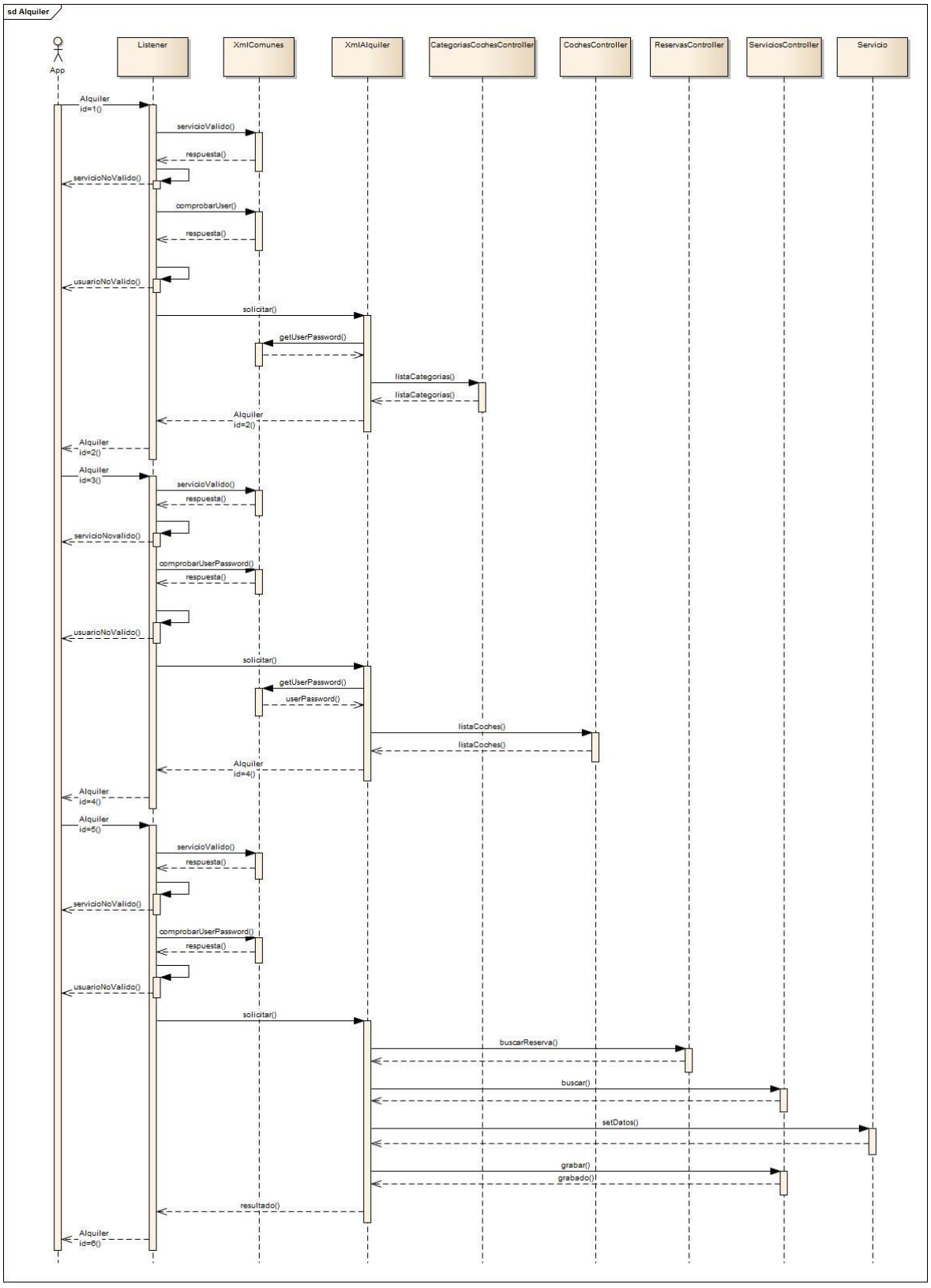


Ilustración 17. Diagrama de secuencia servidor – Servicio de alquiler de coches

4.2.2.11. Historial

Se muestra la interacción entre la aplicación móvil y el servidor web durante la petición del historial. El servlet Listener recibe los datos del cliente y el servicio solicitado. XmlComunes los valida, si todo está correcto Listener pasa la solicitud a XmlHistorial el cual llama al controlador de reservas XmlReservas que se encarga de buscar la reserva del cliente, buscar los datos y devolver el control al servlet con el estado de la operación que será devuelto al cliente.

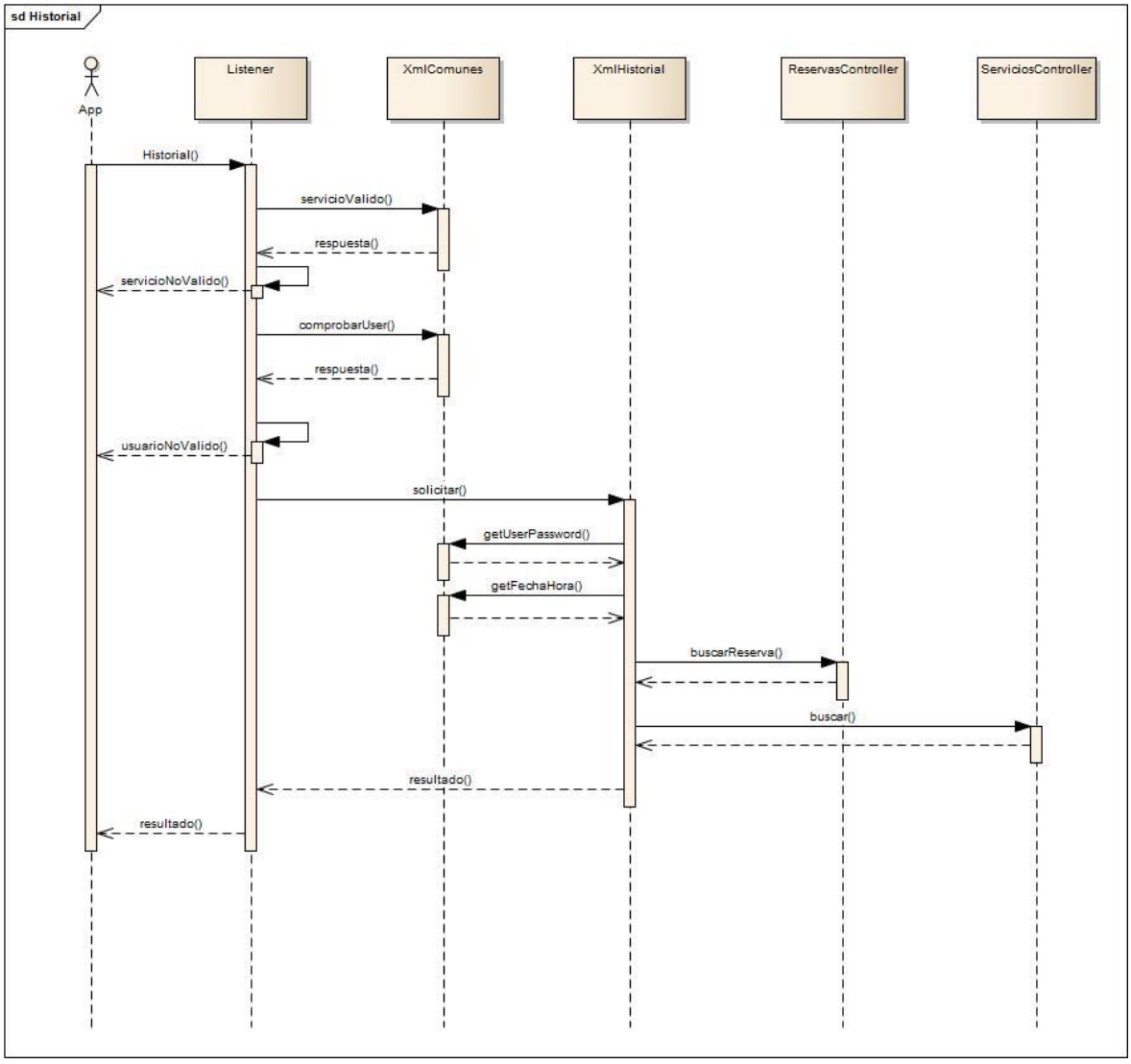


Ilustración 18. Diagrama de secuencia servidor – Historial

4.2.2.12. Cancelar

Se muestra la interacción entre la aplicación móvil y el servidor web durante la petición de cancelación de un servicio. El servlet Listener recibe los datos del cliente y el servicio solicitado. XmlComunes los valida, si todo está correcto Listener pasa la solicitud a XmlCancelar el cual llama al controlador de reservas XmlReservas que se encarga de buscar la reserva del cliente, tramitar la cancelación y devolver el control al servlet con el estado de la operación que será devuelto al cliente.

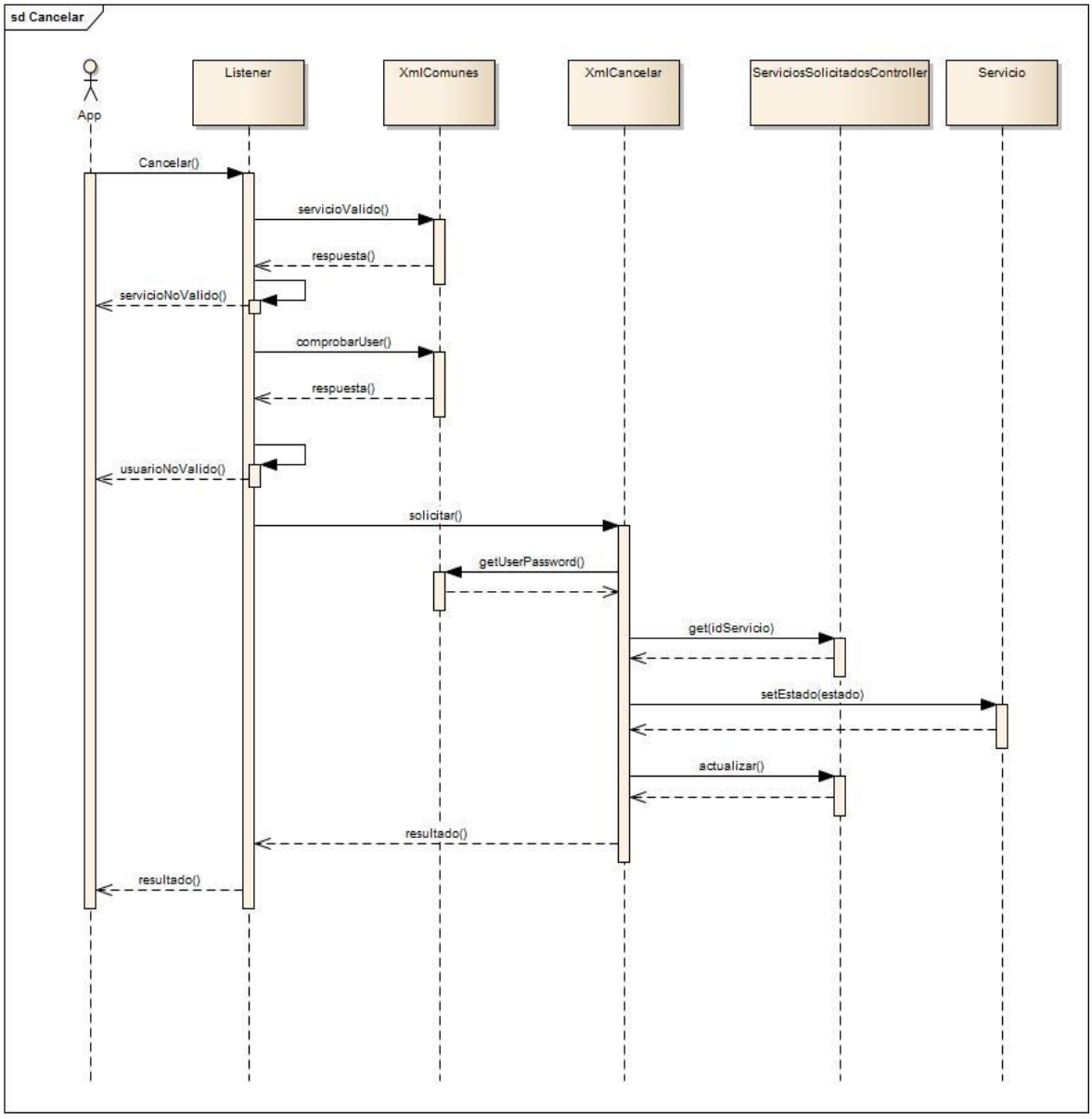


Ilustración 19. Diagrama de secuencia servidor – Cancelar servicio

4.2.2.13 Alrededores

Se muestra la interacción entre la aplicación móvil y el servidor web durante la petición del información sobre lugares próximos al hotel. El servlet Listener recibe los datos del cliente y el servicio solicitado. XmlComunes los valida, si todo está correcto Listener pasa la solicitud a XmlAlrededores el cual llama a AlrededoresController que se encarga de suministrar los datos, se devuelve el control al servlet con el estado de la operación que será devuelto al cliente.

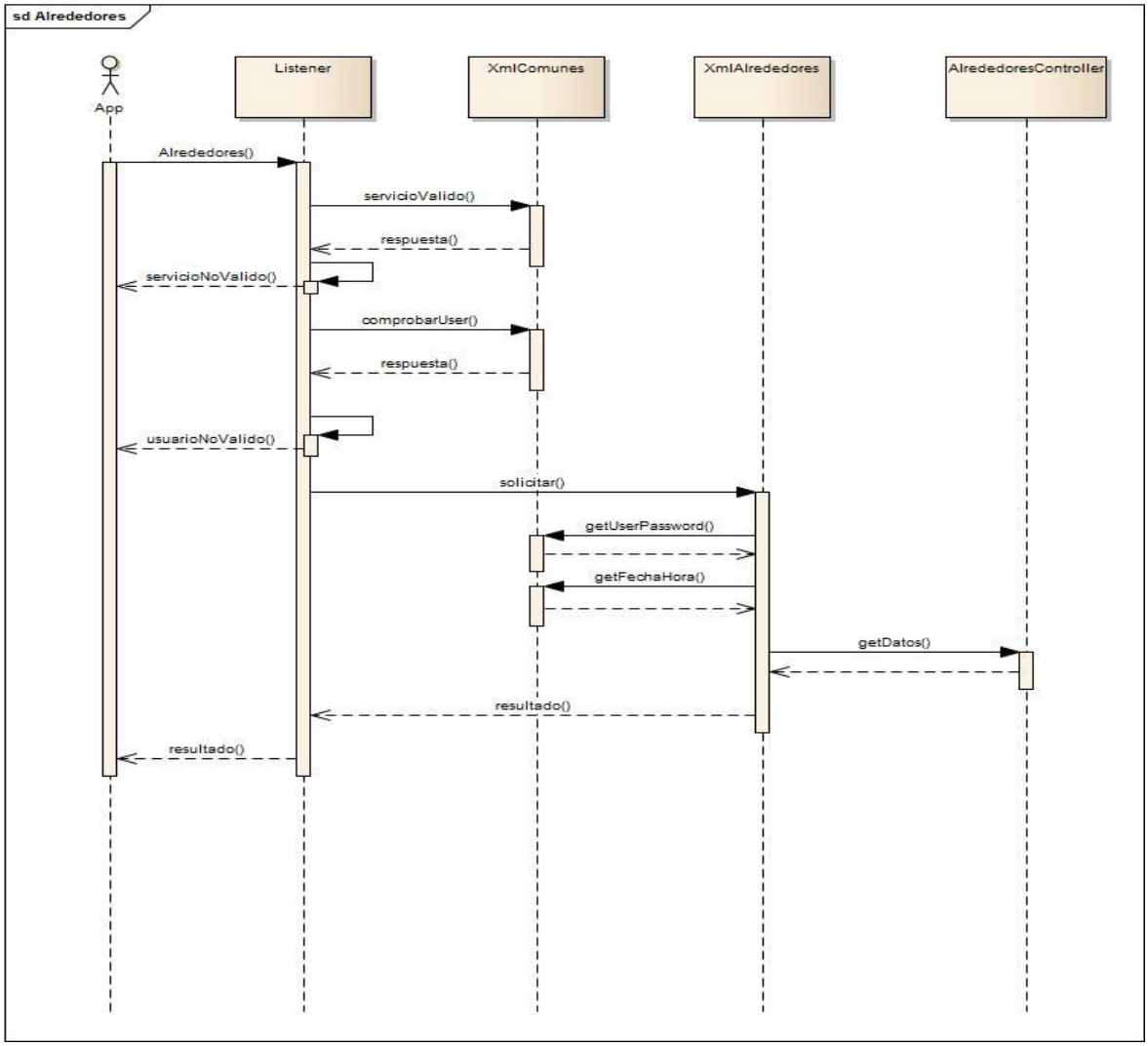


Ilustración 20. Diagrama de secuencia servidor – Alrededores

4.2.2.14. Checkin

A continuación se muestra el diagrama de secuencia de la operación checkin por parte de los recepcionistas del hotel.

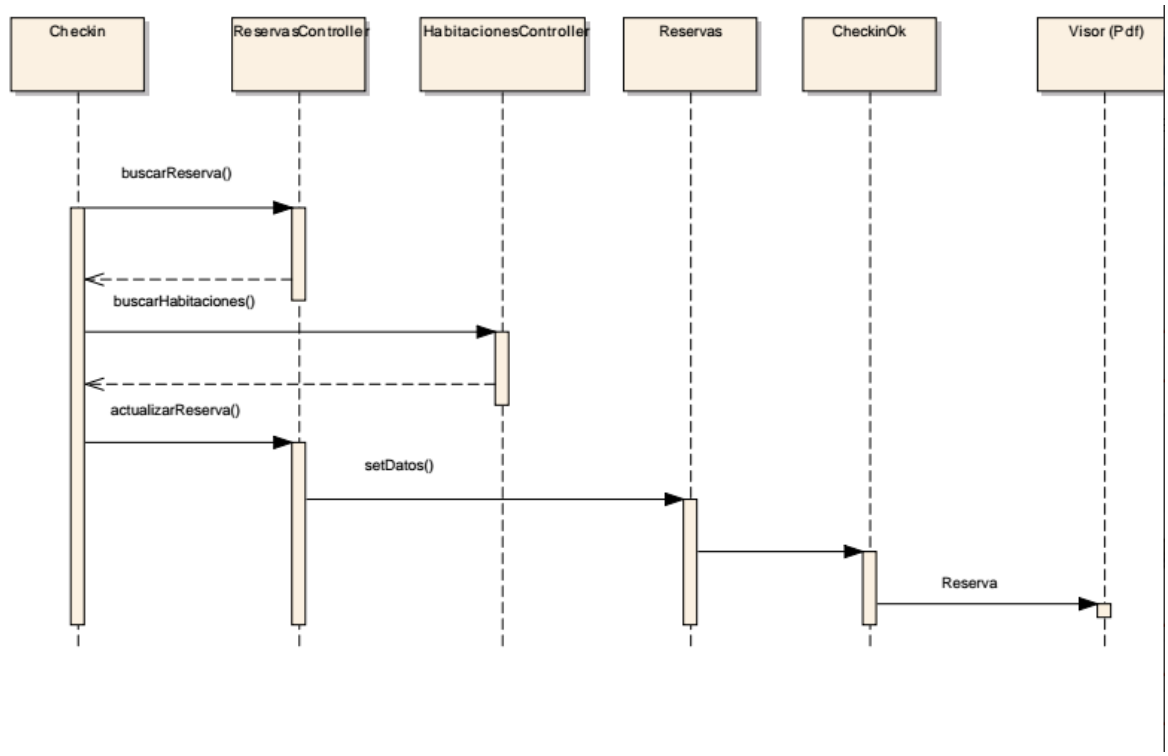


Ilustración 21. Diagrama de secuencia servidor –Checkin

4.2.2.15. Checkout

Análogamente con la operación de checkout.

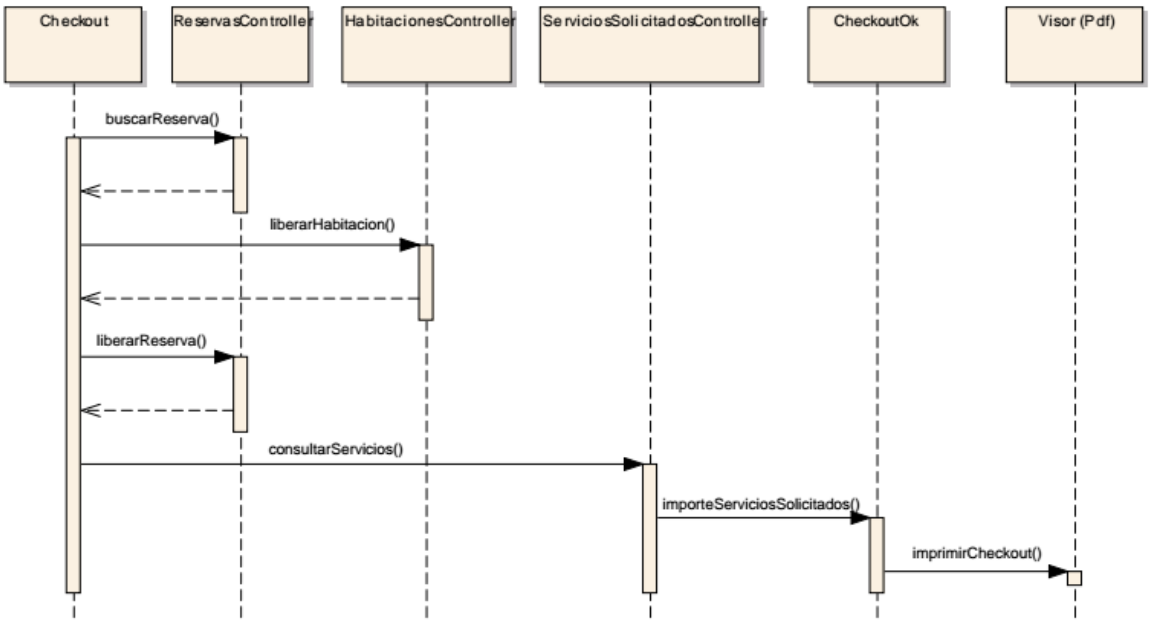


Ilustración 22. Diagrama de secuencia servidor –Checkout

4.3. Aplicación Android

4.3.1. Vista lógica

El concepto de la aplicación ha conducido a la adopción de una arquitectura cliente-servidor, donde los primeros están desarrollados usando la tecnología Android y el segundo utilizando Servlets de Java, siendo la comunicación entre ambos mediante el protocolo HTTP.

Por tanto, este proyecto está fundamentado en la comunicación clientes-servidor, en ambos sentidos, aún siendo las de sentido servidor-cliente respuestas a las peticiones de éstos.

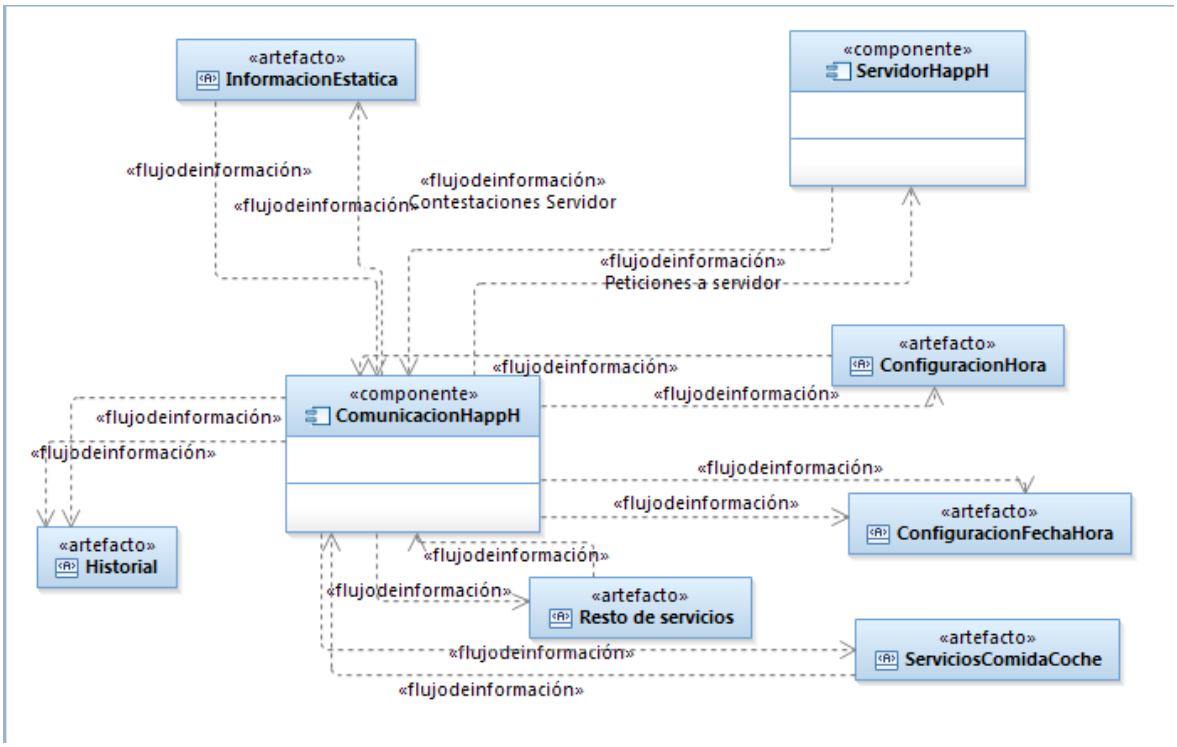


Ilustración 23. Diagrama de componentes de la aplicación Android

4.3.2. Componentes del diseño arquitectónico

La aplicación Android consta de cuatro paquetes de clases, que diferencian claramente la funcionalidad implementada en cada uno de ellos. Los paquetes constituyentes de la aplicación son:

- **com.es.happh**: es el paquete principal de la aplicación donde se implementan las funciones principales relativas a
- **com.es.clasesInterfaz:**paquete que contiene la clase que define la interfaz que sirve para mostrar cada servicio en la pantalla principal.
- **com.es.comunicacionXML:** es el paquete que contiene las clases relativas a la comunicación y creación-traducción de los mensajes XML intercambiados entre la aplicación Android y el servidor.
- **com.es.ficheroDatos:** este paquete incluye únicamente el fichero que contiene los datos de nombre de usuario y contraseña para el acceso a los servicios.

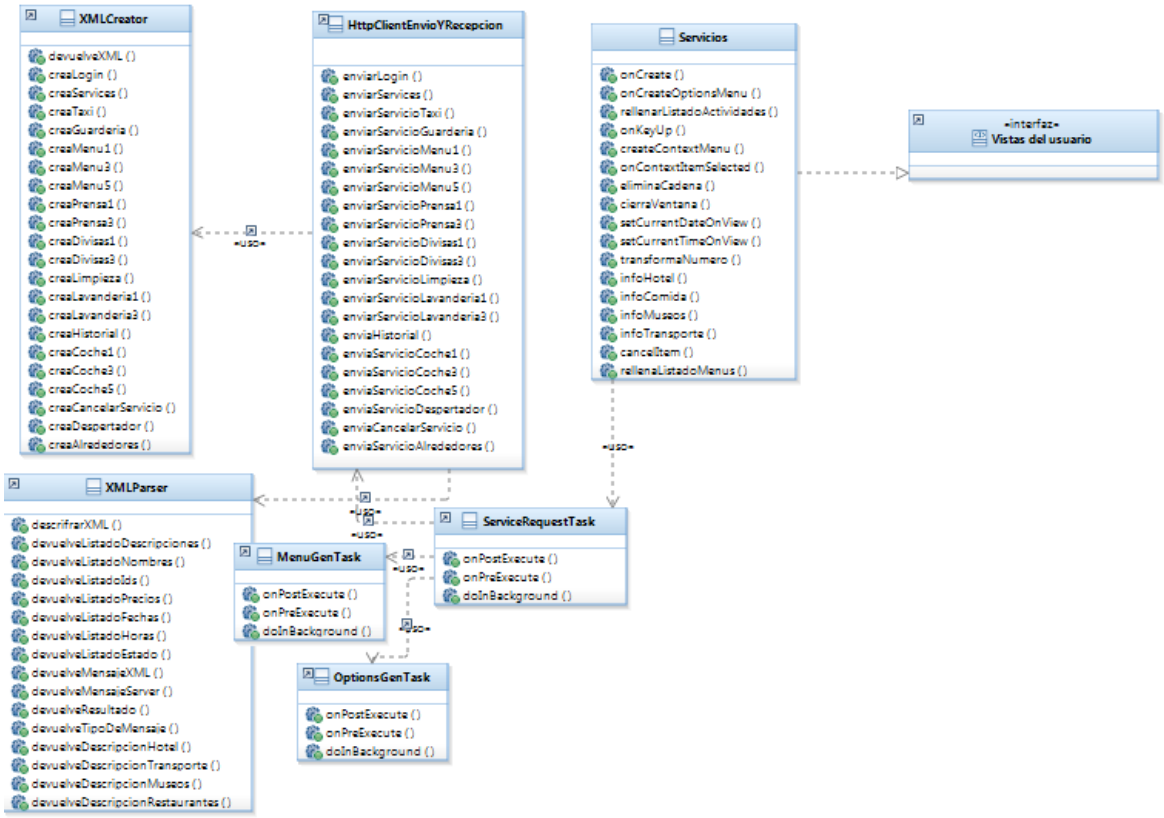


Ilustración 24.Diagrama de componentes

4.3.3. Dominio de clases

En este apartado se hará referencia al modelo de datos de dominio, de cómo están estructuradas las clases que utiliza el sistema, con sus métodos y atributos, relaciones entre ellas y dependencias.

Por ser el diagrama resultante de grandes dimensiones, se mostrará dividido en diversos diagramas de menor tamaño.

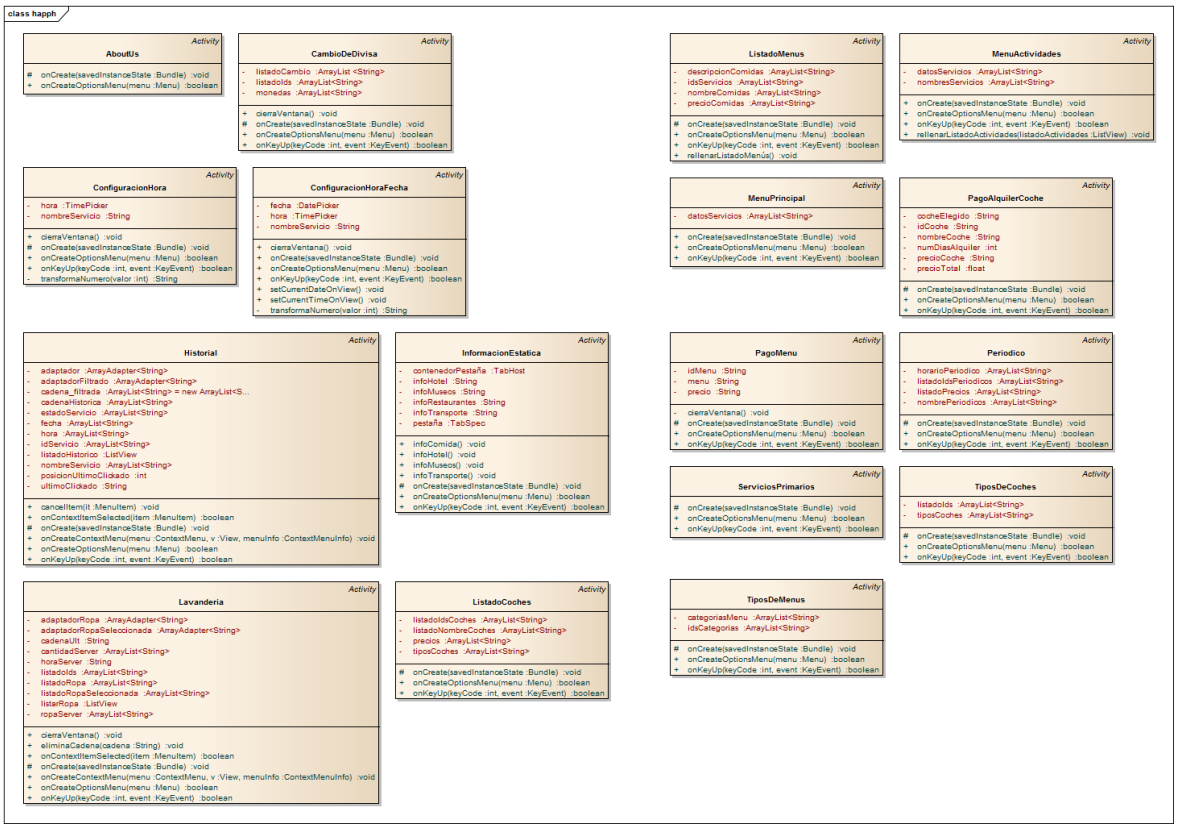


Ilustración 254.Paquete com.es.happh

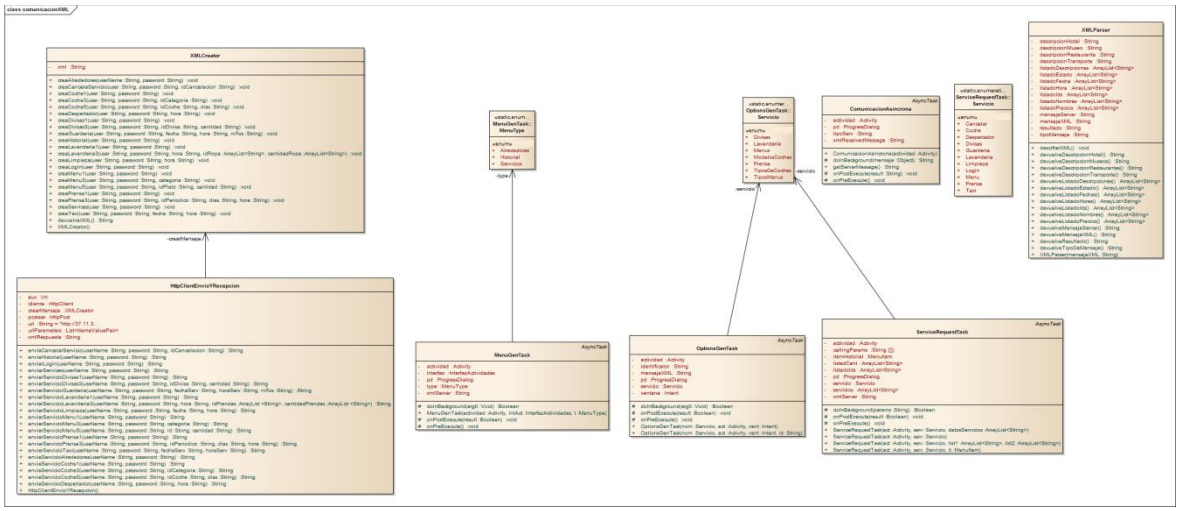


Ilustración 26.Paquete com.es.comunicacionXML

Ilustración 26.Paquete com.es.clasesInterfaz

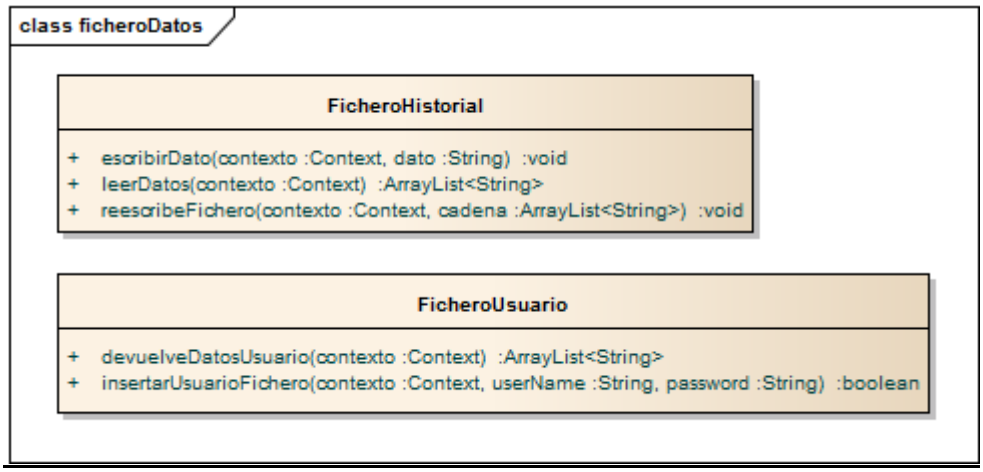


Ilustración 277.Paquete com.es.ficheroDatos

4.3.4. Vista de interacción

En este apartado se presentan los diagramas de secuencia que muestran el comportamiento de la aplicación en su parte Android.

4.3.4.1. Login

Se muestra la interacción entre componentes durante la operación de login en el sistema, para lo cual el cliente deberá introducir su nombre de usuario y contraseña, facilitados previamente, y pulsar el botón “Aceptar” para iniciar el proceso.

En este caso se creará un objeto del tipo ServiceRequestTask que implementa la comunicación asíncrona que permite la notificación simultánea al usuario del proceso de la misma, mediante un diálogo modal. Este objeto hace uso en primer lugar de la clase HttpClientEnvioYrecepcion, donde se explicitan las llamadas a métodos de DefaultHttpClient, que realizarán finalmente la comunicación con el servidor mediante una llamada HTTP con la cadena de texto XML formateada del modo adecuado.

Si la comunicación es exitosa, se creará un XMLParser que interprete la respuesta obtenida del servidor y, si el login se ha producido y así es comunicado en el mensaje XML de vuelta, se creará un FicheroUsuario para almacenar las credenciales del cliente, mostrándose la ventana de servicios primarios correspondiente.

En cualquier caso se notificará al usuario el éxito o no en el procedimiento de login mediante un mensaje de tipo Toast, distinguiendo si el error es debido a un problema de conexión o de credenciales erróneas.

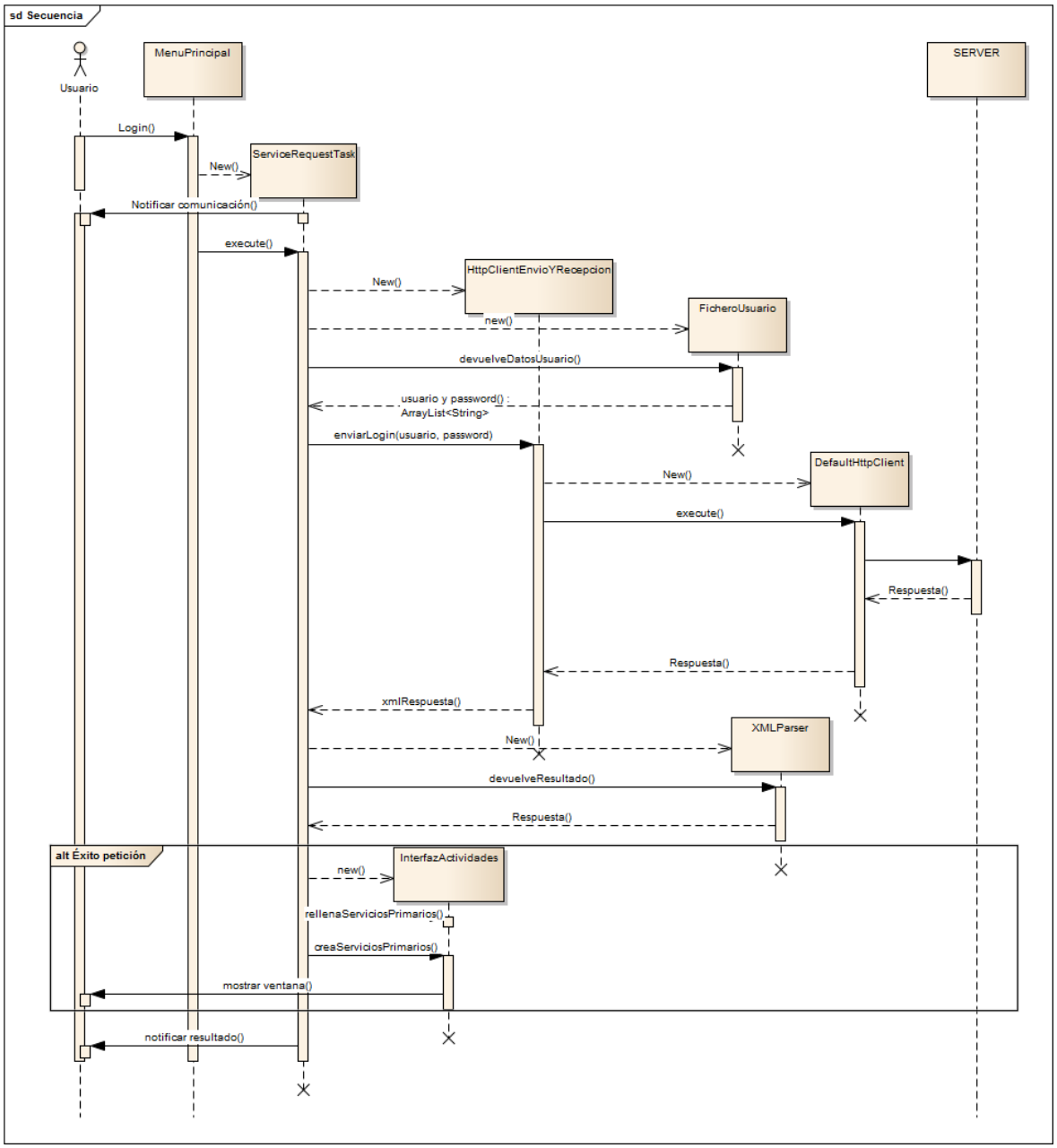


Ilustración 28. Diagrama de secuencia aplicación –Login

4.3.4.2 Alrededores

Muestra la interacción de los distintos componentes durante la petición del servicio de información sobre los alrededores del hotel.

Para la comunicación se crea un objeto de la clase MenuGentask que implementa la comunicación asíncrona que permite notificar al usuario la operación mediante un diálogo de progreso modal. Este objeto hará uso de diversas clases, como FicheroUsuario, para acceder al nombre de usuario y contraseña, y HttpClientEnvioYrecepcion, donde se explicitan las llamadas a métodos de DefaultHttpClient que efectúan finalmente la comunicación con el servidor, enviando una cadena de texto creada en un formato XML definido previamente, mediante una llamada HTTP.

Tras esta, el servidor genera una respuesta y si ésta es en los términos establecidos, se requerirá de un parser o analizador, implementado en la clase XMLParser, que traduzca el texto XML devuelto por el servidor, a las diversas informaciones según la categoría de las mismas que se mostrarán en sus respectivas pestañas en la nueva ventana de la aplicación.

En caso de error se notificaría al usuario mediante un mensaje modal de tipo Toast indicando tal extremo.

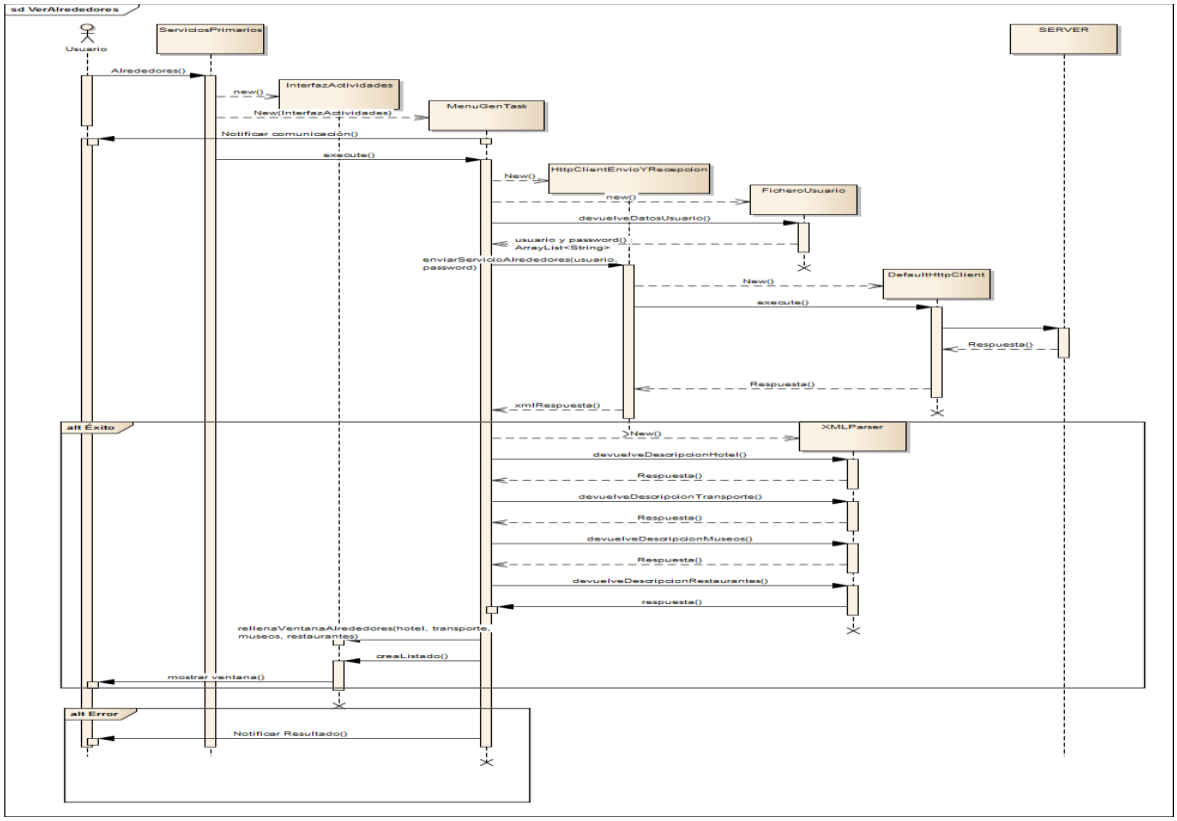


Ilustración 29. Diagrama de secuencia aplicación –Alrededores

4.3.4.3 Servicios

En este diagrama se representan las interacciones entre elementos durante el proceso de solicitud de servicios disponibles para la aplicación, que se inicia cuando el cliente seleccione la opción "Servicios" en la pantalla de servicios primarios.

Tras este punto, se crea un objeto del tipo MenuGenTask, que implementa la comunicación asíncrona y notifica mediante un diálogo modal el progreso de la operación, haciendo uso tanto de la clase FicheroUsuario, para el acceso a las credenciales, como de la clase HttpClientEnvioYRecepcion, donde se explicitan las llamadas a métodos de DefaultHttpClient, que realizarán finalmente la comunicación con el servidor mediante una llamada HTTP con la cadena de texto XML formateada del modo adecuado, requiriendo el servicio de listar los servicios disponibles.

Si la comunicación es exitosa se tomará el mensaje XML devuelto por el servidor y haciendo uso de la clase XMLParser se traducirá a un listado nombres y descripciones de los servicios disponibles que se mostrará en una nueva ventana creada al efecto en la aplicación.

De producirse un error se notificaría mediante un mensaje de tipo Toast.

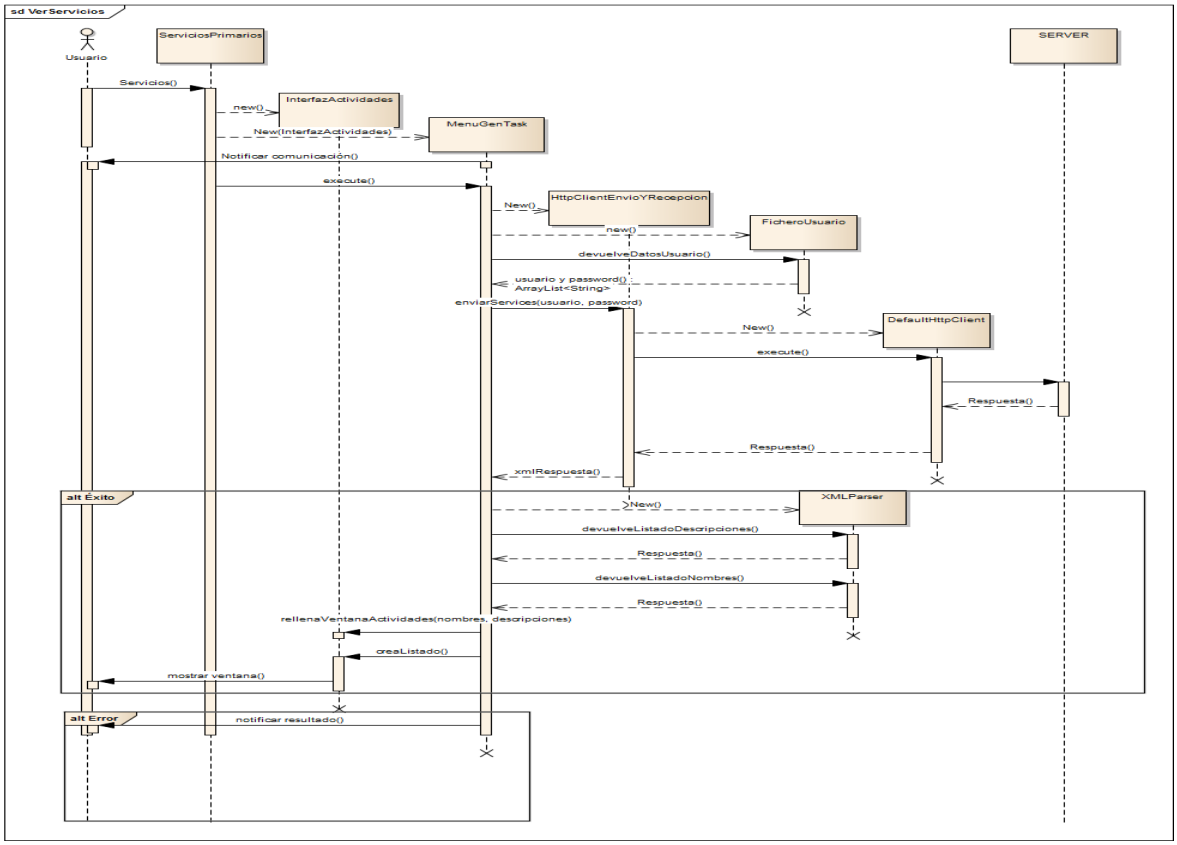


Ilustración 30. Diagrama de secuencia aplicación – Servicios

4.3.4.4. Historial

En este diagrama se muestran las interacciones entre elementos durante el proceso de acceso al historial de servicios pedidos por el usuario, que se inicia al pulsar la opción "Historial" en la pantalla de servicios primarios de la aplicación.

Para la solicitud del historial se crea un objeto de la clase MenuGenTask que notifique la comunicación mediante un diálogo modal simultáneamente a la realización del a comunicación asíncrona con el servidor. Este objeto hará uso de las clases FicheroUsuario, para obtener las credenciales del cliente y HttpClientEnvioYRecepcion, donde se explicitarán las llamadas a métodos de DefaultHttpClient, que realizan finalmente la comunicación con el servidor mediante una llamada HTTP con la cadena de texto XML requiriendo el servicio de historial.

Si la comunicación es exitosa se obtendrá una respuesta en formato XML que, traducida mediante un XMLParser, servirá para mostrar al usuario el listado de servicios solicitados en una nueva ventana creada al efecto.

Si se produce un error se notificará al usuario mediante un mensaje de tipo Toast.

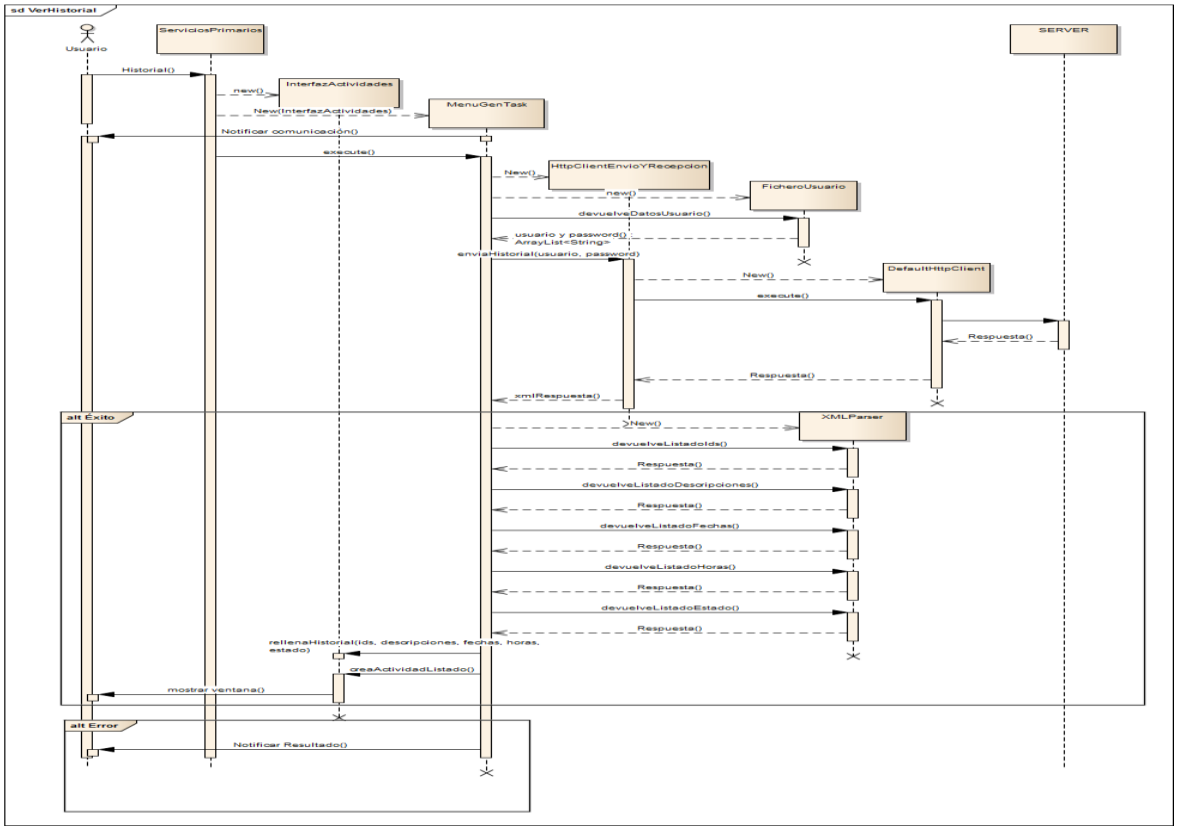


Ilustración 31. Diagrama de secuencia aplicación – Historial

4.3.4.5. Cancelar servicio

En este diagrama se muestran las interacciones entre elementos durante el proceso de solicitud de cancelación de servicios, que se inicia haciendo una pulsación larga sobre el elemento del historial que representa el servicio a cancelar.

Para este procedimiento se crea un objeto del tipo ServiceRequestTask, que implementa la comunicación simultánea con el servidor mientras se notifica al cliente que la operación está en curso mediante un diálogo modal. Este objeto hace uso de las clases FicheroUsuario, para obtener las credenciales del cliente y HttpClientEnvioYRecepcion, donde se explicitarán las llamadas a métodos de DefaultHttpClient, que realizan finalmente la comunicación con el servidor mediante una llamada HTTP con la cadena de texto XML requiriendo el servicio de cancelación.

Si la comunicación es exitosa se obtendrá una respuesta en formato XML que, traducida mediante un XMLParser, servirá para cambiar el color del ítem seleccionado al efecto de notificar al usuario la situación de "cancelado" del servicio al que representa en el historial.

Si se produce un error se notificará al usuario mediante un mensaje de tipo Toast.

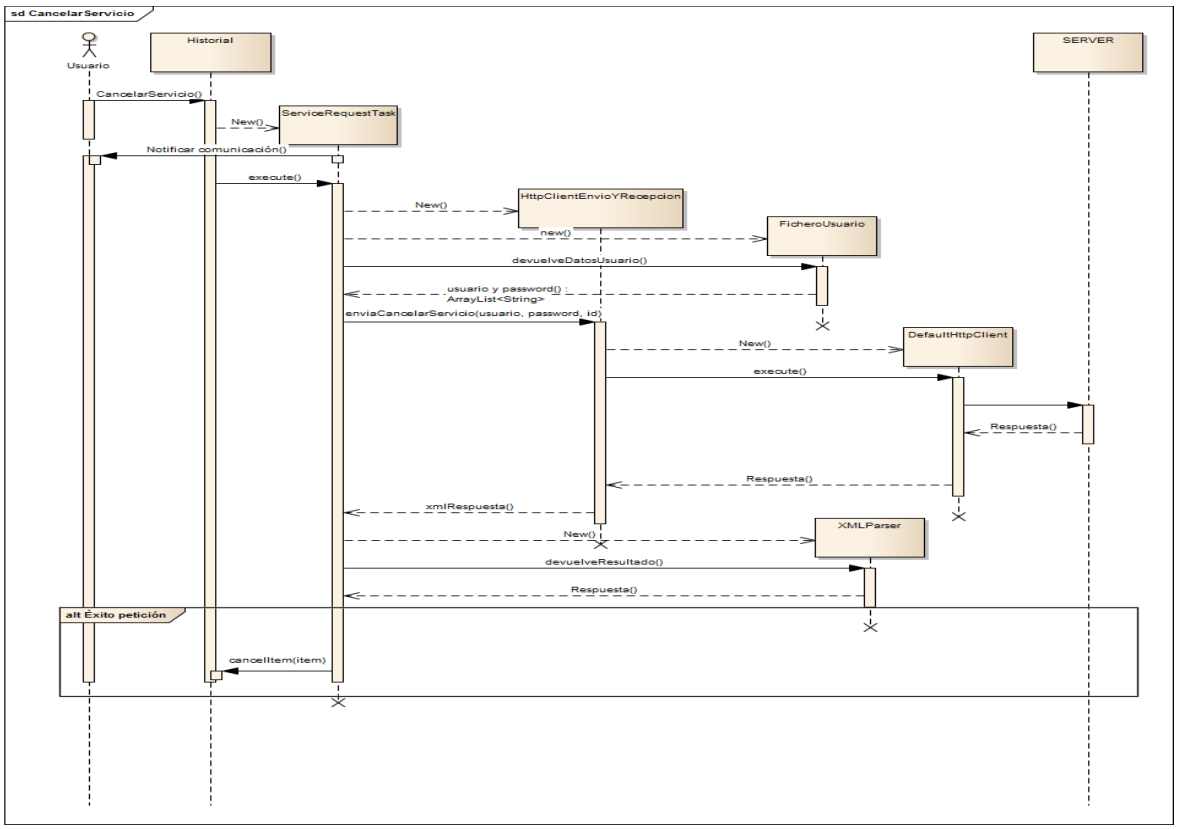


Ilustración 32. Diagrama de secuencia aplicación – Cancelar servicio

4.3.4.6. Servicio de taxi

En este diagrama se muestra el procedimiento de solicitud del servicio de taxi, que se inicia al pulsar sobre el botón "Aceptar" dentro de la pantalla correspondiente, una vez seleccionada la hora y fecha del servicio.

Este servicio hace uso de un objeto de la clase ServiceRequestTask, que notifica al cliente la comunicación mediante un diálogo de progreso modal y efectúa simultáneamente la comunicación con el servidor, enviando la hora y fecha deseada para el servicio. Este objeto dispone de la clase FicheroUsuario, para tomar las credenciales del cliente y de HttpEnvioYRecepcion, para invocar a los métodos correspondientes de la clase DefaultHttpClient que efectuará la invocación de la llamada HTTP con la cadena XML que incluirá el tipo de servicio y las opciones necesarias, que constituye la comunicación con el servidor.

Si la petición surte efecto exitosamente, se notificará al usuario mediante un mensaje modal de tipo Toast el tiempo restante hasta que tenga lugar el servicio o, de ser errónea, el tipo de error ocurrido. Para este punto será necesario hacer uso de la clase XMLParser, que traduzca la respuesta XML recibida a los términos adecuados.

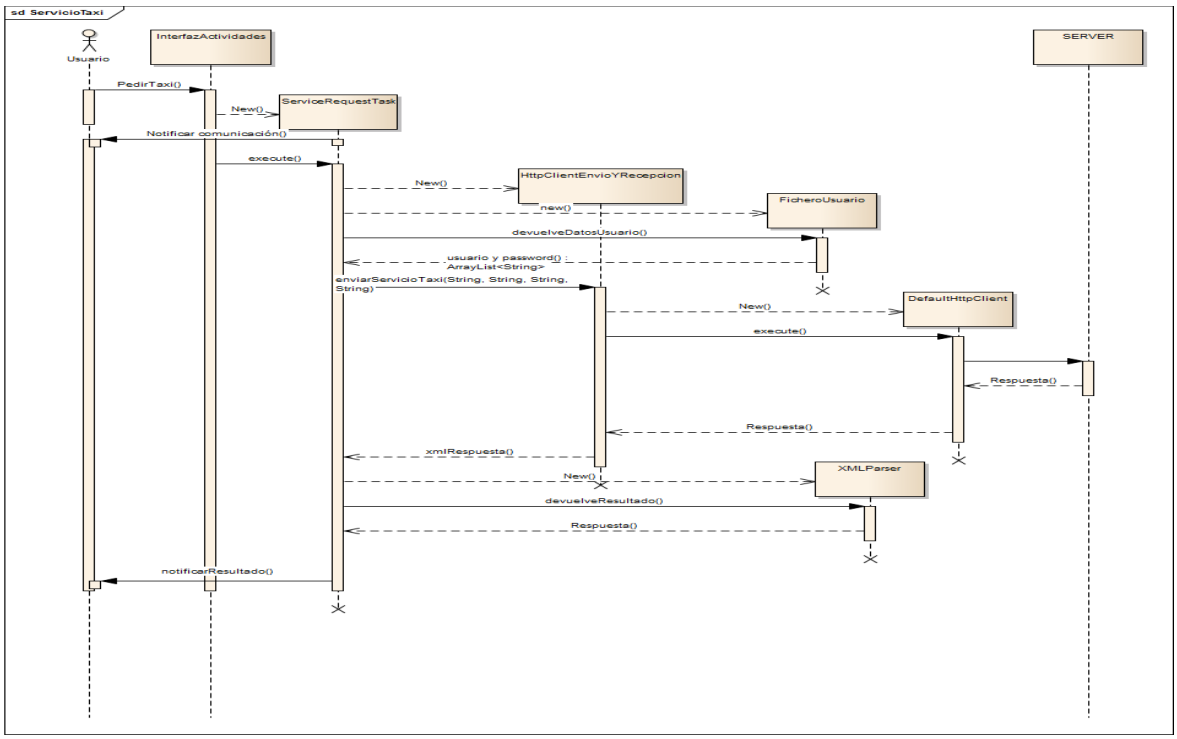


Ilustración 33. Diagrama de secuencia aplicación – Servicio de taxi

4.3.4.7. Servicio de limpieza

En este diagrama se muestra el procedimiento de solicitud del servicio de limpieza, que se inicia al pulsar sobre el botón "Aceptar" dentro de la pantalla correspondiente, una vez seleccionada la hora del servicio.

Este servicio hace uso de un objeto de la clase ServiceRequestTask, que notifica al cliente la comunicación mediante un diálogo de progreso modal y efectúa simultáneamente la comunicación con el servidor, enviando la hora deseada para el servicio. Este objeto dispone de la clase FicheroUsuario, para tomar las credenciales del cliente y de HttpEnvioYRecepcion, para invocar a los métodos correspondientes de la clase DefaultHttpClient que efectuará la invocación de la llamada HTTP con el mensaje XML formateado incluyendo el tipo de servicio y las opciones necesarias, que integra la comunicación con el servidor.

Si la petición surte efecto exitosamente, se notificará al usuario mediante un mensaje modal de tipo Toast el tiempo restante hasta que tenga lugar el servicio o, de ser errónea, el tipo de error ocurrido, siendo para este punto necesaria la clase XMLParser que traduzca la respuesta XML a los términos correspondientes, éxito o no.

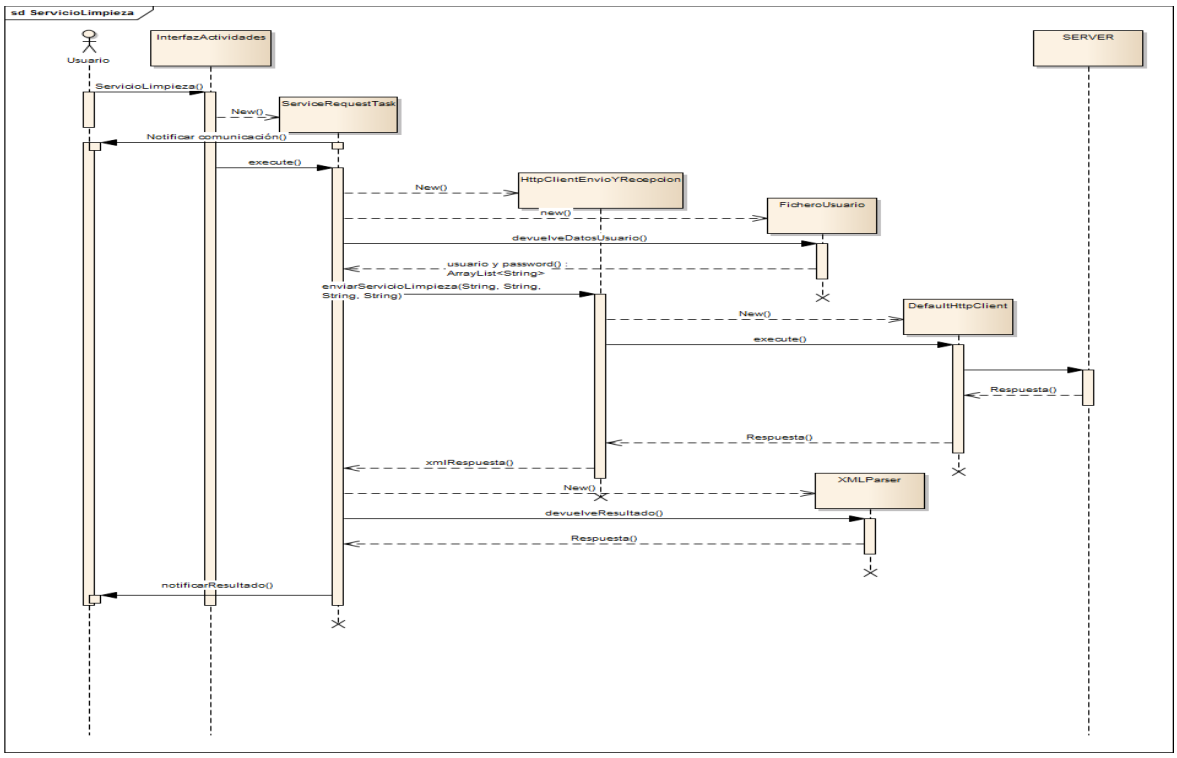


Ilustración 34. Diagrama de secuencia aplicación – Servicio de limpieza

4.3.4.8. Servicio despertador

En este diagrama se muestra el procedimiento de solicitud del servicio de despertador, que se inicia al pulsar sobre el botón "Aceptar" dentro de la pantalla correspondiente, habiendo seleccionado la hora deseada.

Este servicio hace uso de un objeto de la clase ServiceRequestTask, que notifica al cliente la comunicación mediante un diálogo de progreso modal y efectúa simultáneamente la comunicación con el servidor, enviando la hora deseada para el servicio. Este objeto dispone de la clase FicheroUsuario, para tomar las credenciales del cliente y de HttpEnvioYRecepcion, para invocar a los métodos correspondientes de la clase DefaultHttpClient que efectuará la invocación de la llamada HTTP con la cadena XML con las opciones necesarias que constituye la comunicación con el servidor.

Si la petición surte efecto exitosamente, se notificará al usuario mediante un mensaje modal de tipo Toast el tiempo restante hasta que tenga lugar el servicio o, de ser errónea, el tipo de error ocurrido. Para este punto se utilizará la clase XMLParser que traduzca la respuesta XML obtenida del servidor.

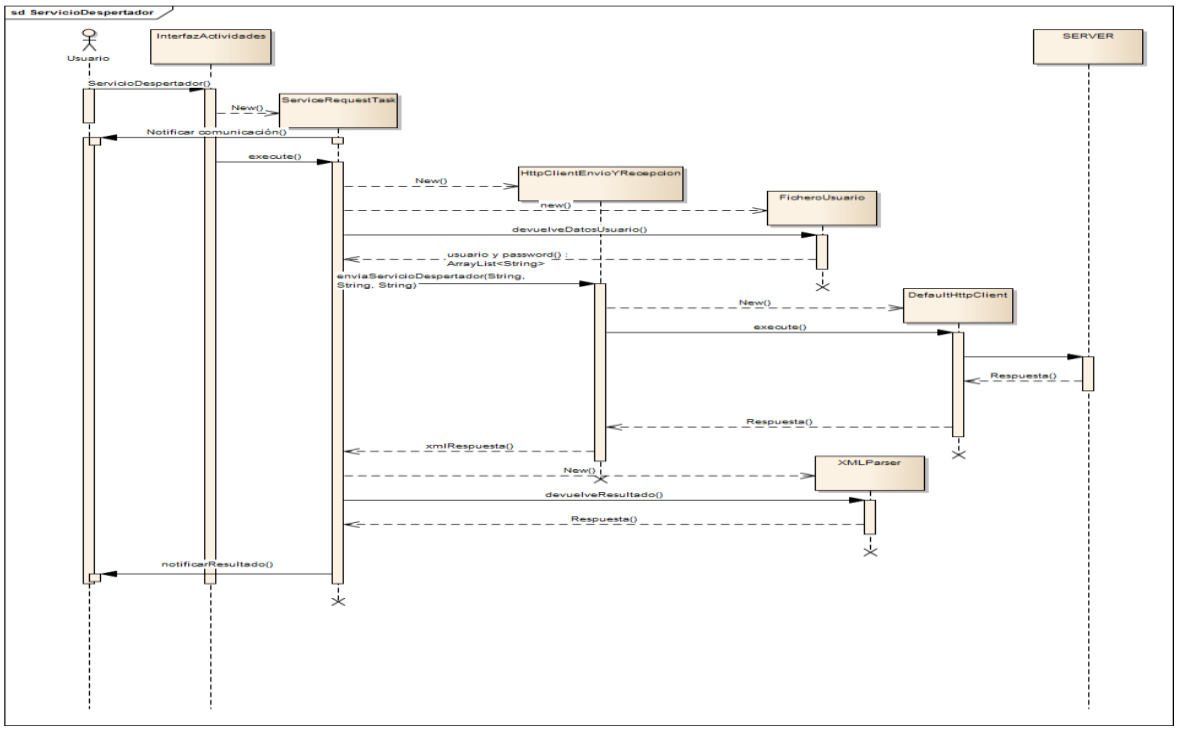


Ilustración 35. Diagrama de secuencia aplicación – Servicio despertador

4.3.4.9. Servicio de prensa

En este diagrama se muestra el procedimiento de solicitud del servicio de prensa, que se compone de dos partes diferenciadas, siendo la primera iniciada al pulsar sobre la opción "Servicio de prensa" del menú de servicios y la segunda tras pulsar el botón "Aceptar" dentro de la pantalla correspondiente, habiendo seleccionado el periódico concreto de entre los ofrecidos, la hora de entrega deseada y si se desea que se repita durante todos los días de la estancia del cliente.

La primera parte del servicio hace uso de la clase OptionsGenTask, que notifica la comunicación con el servidor al tiempo que realiza la misma, enviando la identificación del servicio, hace uso de la clase FicheroUsuario para tomar las credenciales del cliente y de HttpEnvioYRecepcion, para invocar a los métodos correspondientes de la clase DefaultHttpClient que efectuará la invocación de la llamada HTTP con la cadena XML con la identificación de servicio necesaria que constituye la comunicación con el servidor.

La respuesta obtenida será traducida mediante XMLParser para listar los periódicos disponibles para los clientes.

Tras este paso, el siguiente es opcional y tendrá lugar si el cliente presiona el botón aceptar, una vez seleccionadas las opciones citadas con anterioridad.

Este servicio hace uso de un objeto de la clase ServiceRequestTask, que notifica al cliente la comunicación mediante un diálogo de progreso modal y efectúa simultáneamente la comunicación con el servidor, enviando la identificación del servicio, el periódico y hora de entrega deseada para el servicio. Este objeto dispone de la clase FicheroUsuario, para tomar las credenciales del cliente y de HttpEnvioYRecepcion, para invocar a los métodos correspondientes de la clase DefaultHttpClient que efectuará la invocación de la llamada HTTP con la cadena XML con las opciones necesarias que constituye la comunicación con el servidor.

Si la petición surte efecto exitosamente, se notificará al usuario mediante un mensaje modal de tipo Toast el tiempo restante hasta que tenga lugar el servicio o, de ser errónea, el tipo de error ocurrido. Para este punto se utilizará la clase XMLParser que traduzca la respuesta XML obtenida del servidor a los términos éxito o no.

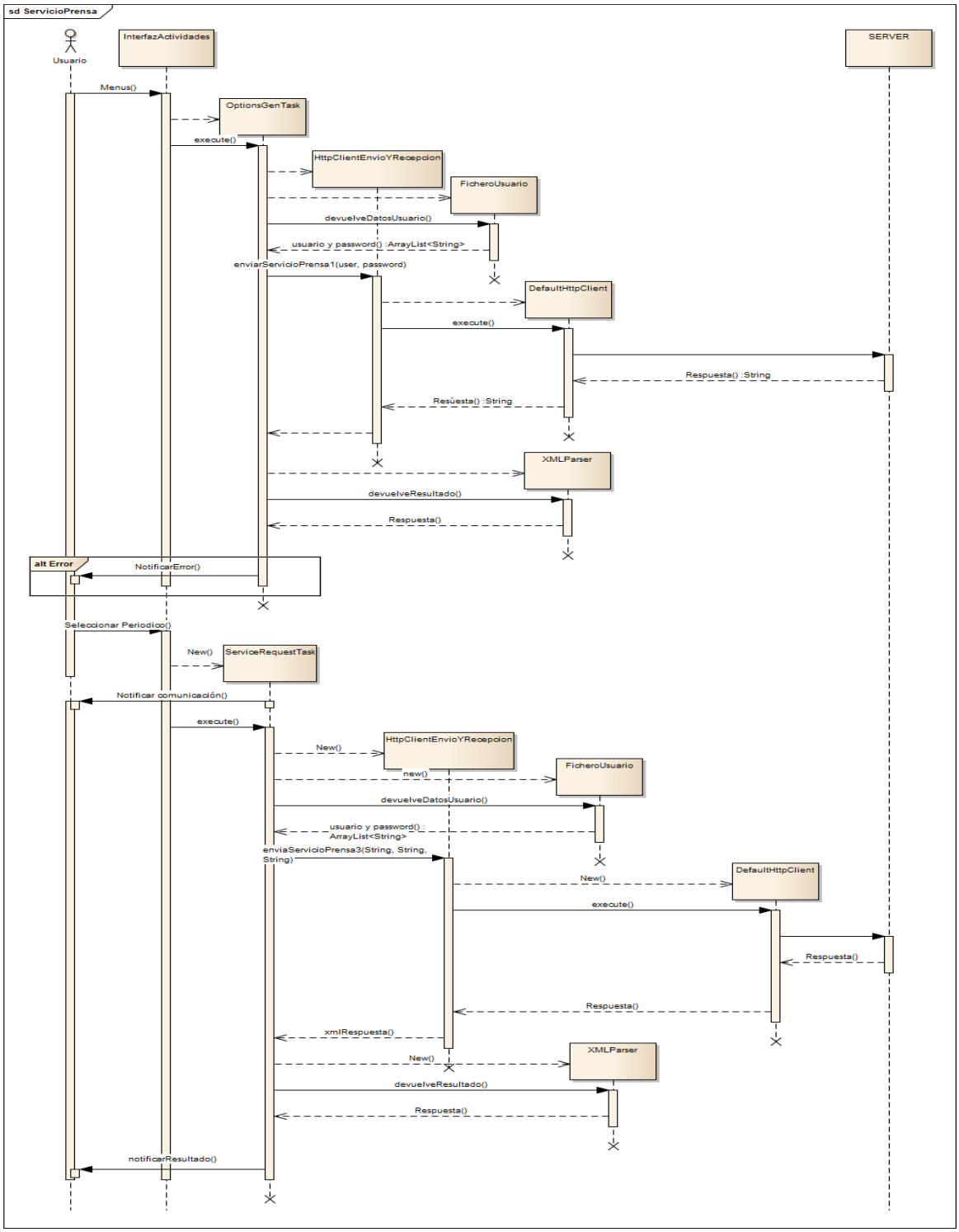


Ilustración 36. Diagrama de secuencia aplicación – Servicio de prensa

4.3.4.10. Servicio de lavandería

En este diagrama se muestra el procedimiento de solicitud del servicio de lavandería, que se compone de dos partes diferenciadas, siendo la primera iniciada al pulsar sobre la opción "Lavandería" del menú de servicios y la segunda tras pulsar el botón "Aceptar" dentro de la pantalla correspondiente, habiendo listado las prendas a enviar junto a sus cantidades, así como la hora para recogida de las mismas.

La primera parte del servicio hace uso de la clase OptionsGenTask, que notifica la comunicación con el servidor al tiempo que realiza la misma, enviando la identificación del servicio, hace uso de la clase FicheroUsuario para tomar las credenciales del cliente y de HttpEnvioYRecepcion, para invocar a los métodos correspondientes de la clase DefaultHttpClient que efectuará la invocación de la llamada HTTP con la cadena XML con la identificación de servicio necesaria que constituye la comunicación con el servidor.

La respuesta obtenida será traducida mediante XMLParser para listar los tipos de prendas disponibles para los clientes, salvo que se hubiera producido un error, siendo notificado mediante un mensaje modal de tipo Toast.

Tras este paso, el siguiente es opcional y tendrá lugar si el cliente presiona el botón aceptar, una vez seleccionadas las opciones citadas con anterioridad.

Este servicio hace uso de un objeto de la clase ServiceRequestTask, que notifica al cliente la comunicación mediante un diálogo de progreso modal y efectúa simultáneamente la comunicación con el servidor, enviando la identificación del servicio, el listado con las prendas y cantidades respectivas, y hora de entrega deseada para el servicio. Este objeto dispone de la clase FicheroUsuario, para tomar las credenciales del cliente y de HttpEnvioYRecepcion, para invocar a los métodos correspondientes de la clase DefaultHttpClient que efectuará la invocación de la llamada HTTP con la cadena XML con las opciones necesarias que constituye la comunicación con el servidor.

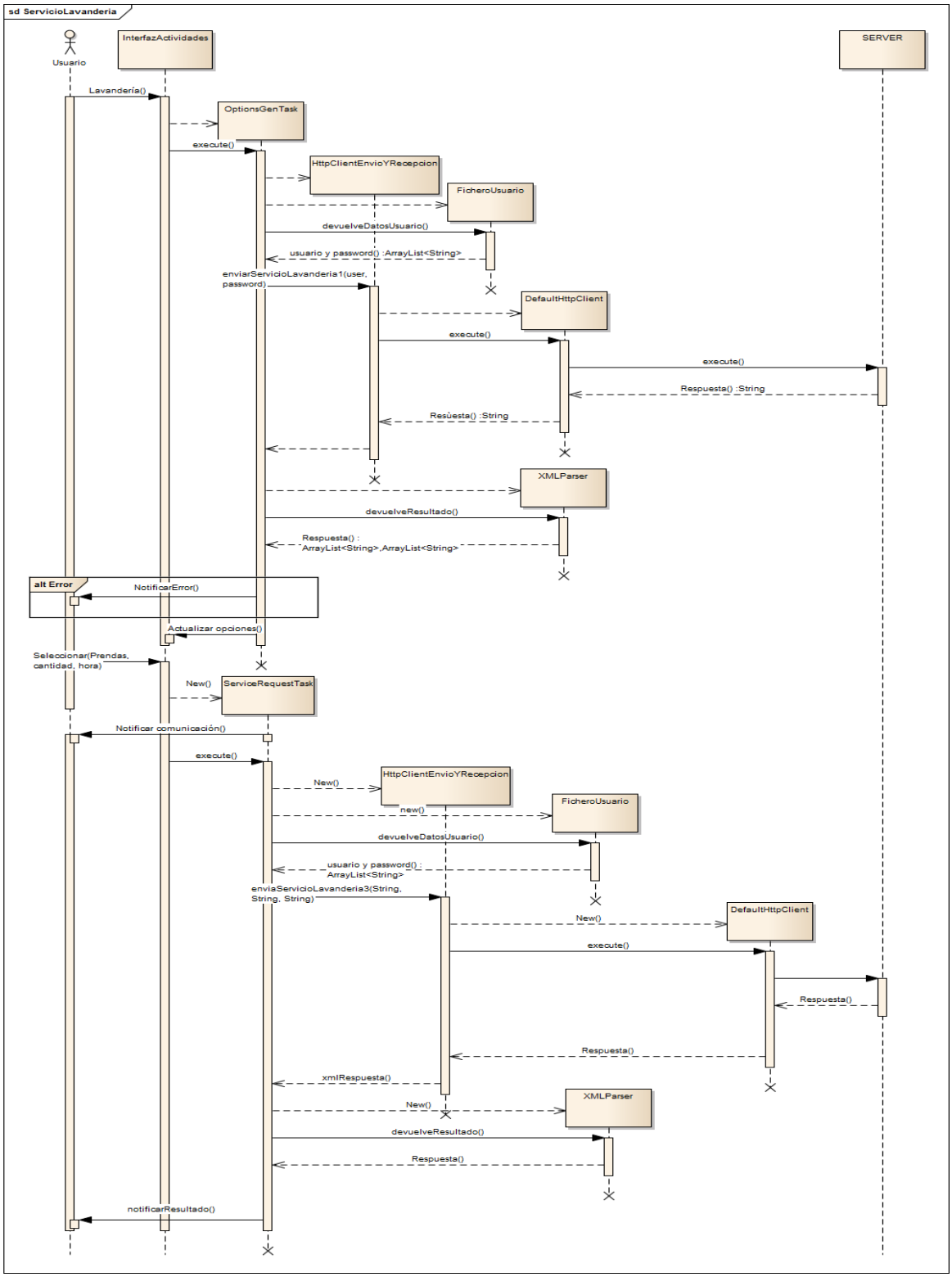


Ilustración 37. Diagrama de secuencia aplicación – Servicio de lavandería

4.3.4.11. Servicio de divisas

En este diagrama se muestra el procedimiento de solicitud del servicio de cambio de divisas, que se compone de dos partes diferenciadas, siendo la primera iniciada al pulsar sobre la opción "Cambio de divisas" del menú de servicios y la segunda tras pulsar el botón "Aceptar" dentro de la pantalla correspondiente, habiendo seleccionado la moneda deseada y la cantidad a cambiar.

La primera parte del servicio hace uso de la clase OptionsGenTask, que notifica la comunicación con el servidor al tiempo que realiza la misma, enviando la identificación del servicio, hace uso de la clase FicheroUsuario para tomar las credenciales del cliente y de HttpEnvioYRecepcion, para invocar a los métodos correspondientes de la clase DefaultHttpClient que efectuará la invocación de la llamada HTTP con la cadena XML con la identificación de servicio necesaria que constituye la comunicación con el servidor.

La respuesta obtenida será traducida mediante XMLParser para listar las distintas monedas disponibles para los clientes, salvo que se hubiera producido un error, que se notificaría oportunamente.

Tras este paso, el siguiente es opcional y tendrá lugar si el cliente presiona el botón aceptar, una vez seleccionadas las opciones citadas con anterioridad.

Este servicio hace uso de un objeto de la clase ServiceRequestTask, que notifica al cliente la comunicación mediante un diálogo de progreso modal y efectúa simultáneamente la comunicación con el servidor, enviando la identificación del servicio, la moneda a cambiar y la cantidad deseada. Este objeto dispone de la clase FicheroUsuario, para tomar las credenciales del cliente y de HttpEnvioYRecepcion, para invocar a los métodos correspondientes de la clase DefaultHttpClient que efectuará la invocación de la llamada HTTP con la cadena XML con las opciones necesarias que constituye la comunicación con el servidor.

Si la petición surte efecto exitosamente, se notificará al usuario mediante un mensaje modal de tipo Toast el tiempo restante hasta que tenga lugar el servicio o, de ser errónea, el tipo de error ocurrido. Para este punto se utilizará la clase XMLParser que traduzca la respuesta XML obtenida del servidor, a los términos éxito o no.

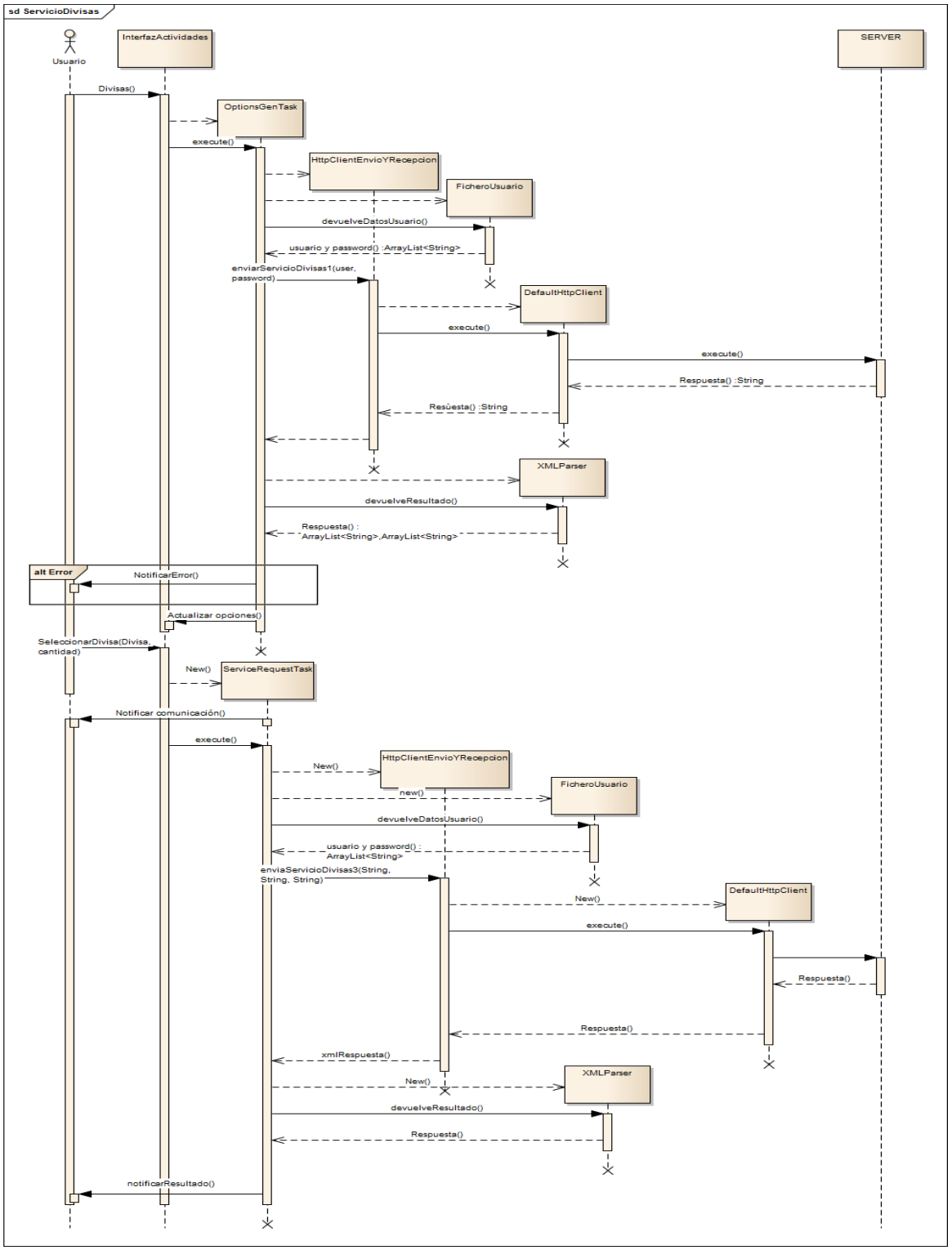


Ilustración 38. Diagrama de secuencia aplicación – Servicio de cambio de divisas

4.3.4.12. Servicio de comidas

En este diagrama se muestra el procedimiento de solicitud del servicio de cambio de divisas, que se compone de tres partes diferenciadas, siendo la primera iniciada al pulsar sobre la opción "Servicio de comidas" del menú de servicios, la segunda tras seleccionar la categoría deseada y la tercera tras pulsar el botón "Aceptar" dentro de la pantalla correspondiente, habiendo indicado la cantidad deseada para el ítem seleccionado.

Las partes primera y segunda del servicio hace uso de la clase OptionsGenTask, que notifica la comunicación con el servidor al tiempo que realiza la misma, enviando la identificación del servicio –y la categoría de platos seleccionada en el segundo caso-, hace uso de la clase FicheroUsuario para tomar las credenciales del cliente y de HttpEnvioYRecepcion, para invocar a los métodos correspondientes de la clase DefaultHttpClient que efectuará la invocación de la llamada HTTP con la cadena XML con la identificación de servicio necesaria y las opciones correspondientes que constituye la comunicación con el servidor.

Las respuestas obtenidas serán traducidas mediante XMLParser para listar las distintas categorías de platos disponibles para los clientes, en la primera parte, y las distintas opciones dentro de cada categoría, en la segunda.

Tras este paso, el siguiente es opcional y tendrá lugar si el cliente presiona el botón "Aceptar", una vez seleccionadas las opciones citadas con anterioridad.

Este servicio hace uso de un objeto de la clase ServiceRequestTask, que notifica al cliente la comunicación mediante un diálogo de progreso modal y efectúa simultáneamente la comunicación con el servidor, enviando la identificación del servicio, el plato o ítem seleccionado y la cantidad deseada. Este objeto dispone de la clase FicheroUsuario, para tomar las credenciales del cliente y de HttpEnvioYRecepcion, para invocar a los métodos correspondientes de la clase DefaultHttpClient que efectuará la invocación de la llamada HTTP con la cadena XML con las opciones necesarias que constituye la comunicación con el servidor.

Si la petición surte efecto exitosamente, se notificará al usuario mediante un mensaje modal de tipo Toast el tiempo restante hasta que tenga lugar el servicio o, de ser errónea, el tipo de error ocurrido. Para este punto se utilizará la clase XMLParser que traduzca la respuesta XML obtenida del servidor, a los términos éxito o no.

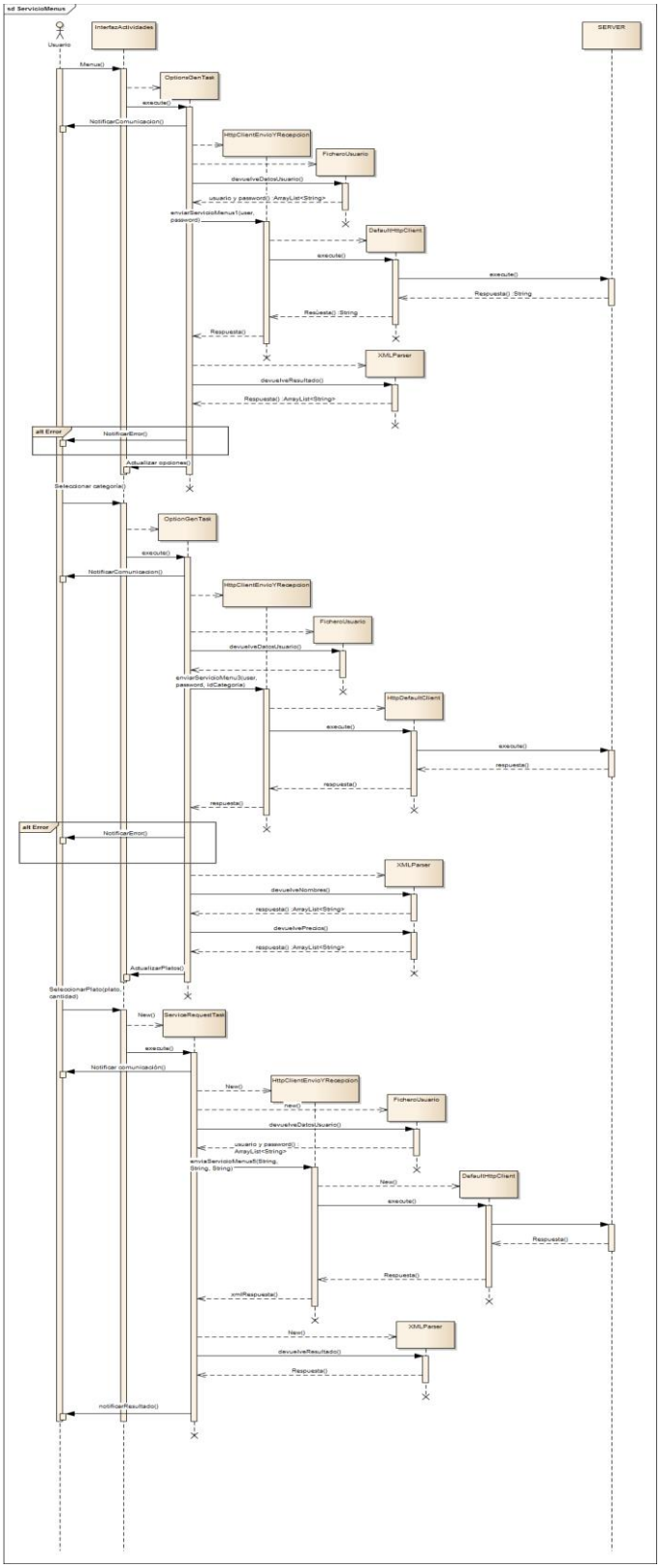


Ilustración 39. Diagrama de secuencia aplicación – Servicio de comidas

4.3.4.13. Servicio de alquiler de coches

En este diagrama se muestra el procedimiento de solicitud del servicio de coches de alquiler, que se compone de tres partes diferenciadas, siendo la primera iniciada al pulsar sobre la opción "Alquiler de coches" del menú de servicios, la segunda tras seleccionar la categoría deseada y la tercera tras pulsar el botón "Aceptar" dentro de la pantalla correspondiente, habiendo indicado el número de días de alquiler deseados para el vehículo seleccionado.

Las partes primera y segunda del servicio hace uso de la clase OptionsGenTask, que notifica la comunicación con el servidor al tiempo que realiza la misma, enviando la identificación del servicio –y la categoría de platos seleccionada en el segundo caso-, hace uso de la clase FicheroUsuario para tomar las credenciales del cliente y de HttpEnvioYRecepcion, para invocar a los métodos correspondientes de la clase DefaultHttpClient que efectuará la invocación de la llamada HTTP con la cadena XML con la identificación de servicio necesaria y las opciones correspondientes que constituye la comunicación con el servidor.

Las respuestas obtenidas serán traducidas mediante XMLParser para listar las distintas categorías de vehículos disponibles para los clientes, en la primera parte, y las distintas opciones dentro de la categoría específica seleccionada, en la segunda.

Tras este paso, el siguiente es opcional y tendrá lugar si el cliente presiona el botón aceptar, una vez seleccionadas las opciones citadas con anterioridad.

Este servicio hace uso de un objeto de la clase ServiceRequestTask, que notifica al cliente la comunicación mediante un diálogo de progreso modal y efectúa simultáneamente la comunicación con el servidor, enviando la identificación del servicio, el plato o ítem seleccionado y la cantidad deseada. Este objeto dispone de la clase FicheroUsuario, para tomar las credenciales del cliente y de HttpEnvioYRecepcion, para invocar a los métodos correspondientes de la clase DefaultHttpClient que efectuará la invocación de la llamada HTTP con la cadena XML con las opciones necesarias que constituye la comunicación con el servidor.

Si la petición surte efecto exitosamente, se notificará al usuario mediante un mensaje modal de tipo Toast el tiempo restante hasta que tenga lugar el servicio o, de ser errónea, el tipo de error ocurrido. Para este punto se utilizará la clase XMLParser que traduzca la respuesta XML obtenida del servidor, a los términos éxito o no.

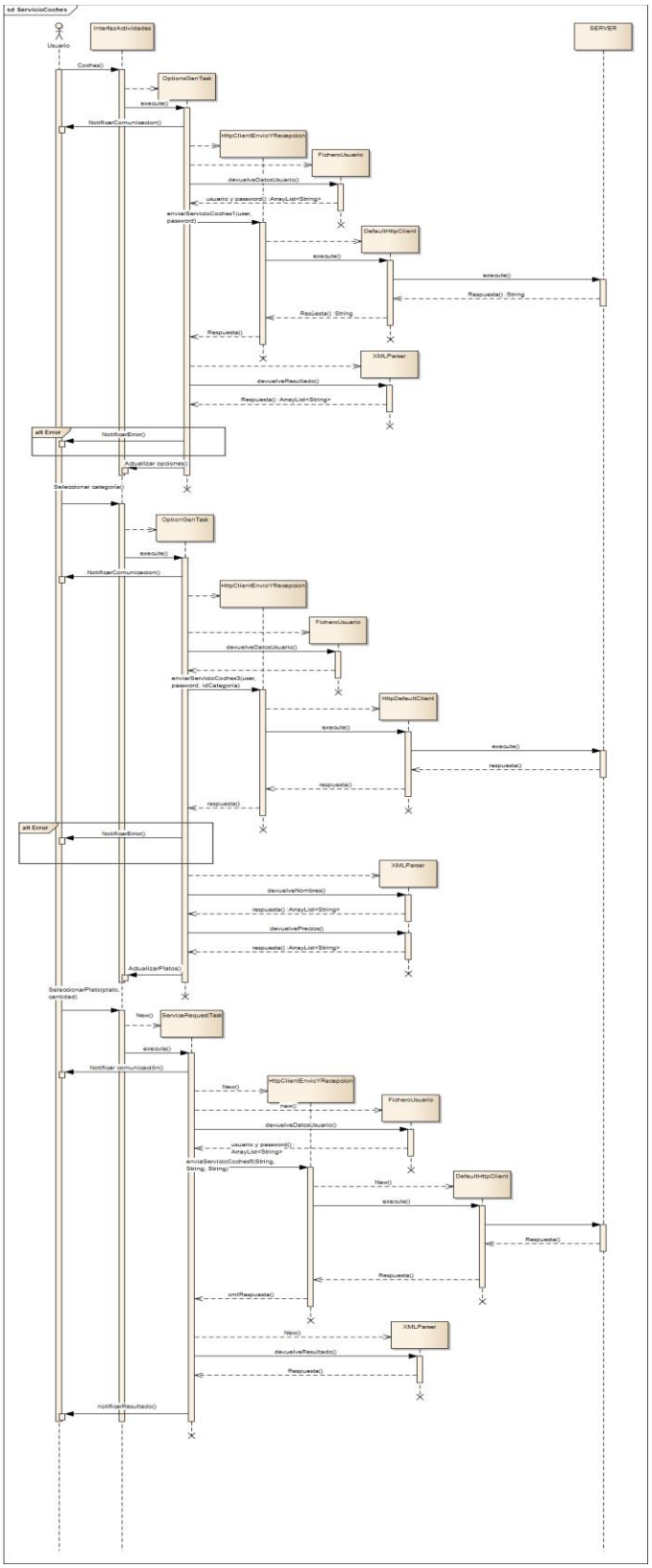


Ilustración 40. Diagrama de secuencia aplicación – Servicio de alquiler de coches

4.3.4.14. Servicio de guardería

En este diagrama se muestra el procedimiento de solicitud del servicio de guardería, que se inicia al pulsar sobre el botón "Aceptar" dentro de la pantalla correspondiente, una vez seleccionada la hora del servicio y el número de niños.

Este servicio hace uso de un objeto de la clase ServiceRequestTask, que notifica al cliente la comunicación mediante un diálogo de progreso modal y efectúa simultáneamente la comunicación con el servidor, enviando la hora deseada para el servicio. Este objeto dispone de la clase FicheroUsuario, para tomar las credenciales del cliente y de HttpEnvioYRecepcion, para invocar a los métodos correspondientes de la clase DefaultHttpClient que efectuará la invocación de la llamada HTTP con el mensaje XML formateado incluyendo el tipo de servicio y las opciones necesarias –hora y número de niños-, que integra la comunicación con el servidor.

Si la petición surte efecto exitosamente, se notificará al usuario mediante un mensaje modal de tipo Toast el tiempo restante hasta que tenga lugar el servicio o, de ser errónea, el tipo de error ocurrido, siendo para este punto necesaria la clase XMLParser que traduzca la respuesta XML a los términos correspondientes, éxito o no.

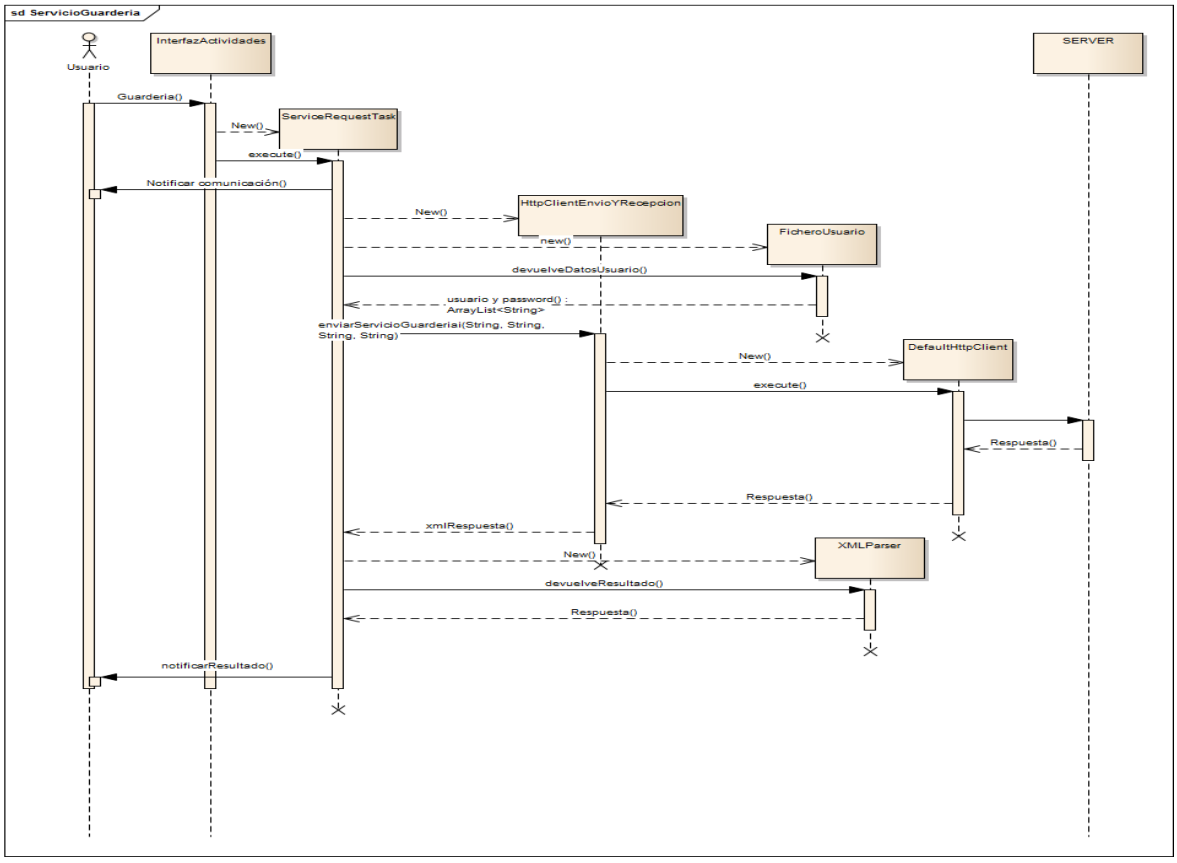


Ilustración 41. . Diagrama de secuencia aplicación – Servicio de guardería

4.4. Diagrama de despliegue

A continuación se muestra el diagrama de despliegue de todo el sistema en su conjunto.

El cliente puede solicitar los servicios del hotel a través de dos vías

- Aplicación web: la capa de presentación se basa en HTML 5 y CSS 3, la capa de negocio se implementa en con servlets JEE. Para acceder a ella se utiliza un navegador web
- Aplicación Android: se necesita un Smartphone Android. A través de HttpClient se realizan las peticiones al servlet Listener del servidor web, el cual se encarga de dialogar con el cliente, cargar las clases necesarias e invocar a sus métodos para gestionar los servicios que demandan los clientes.

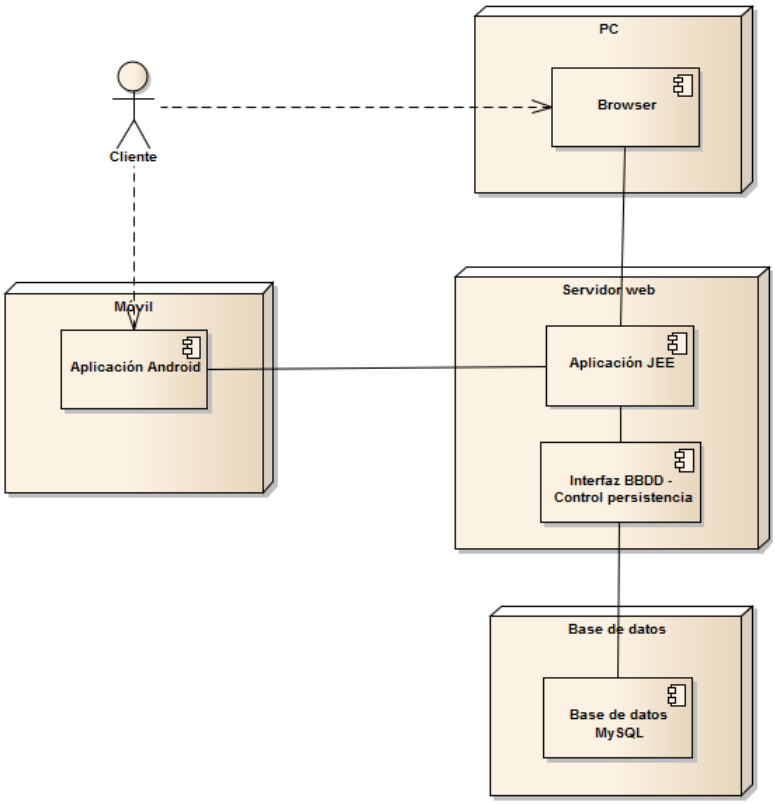


Ilustración 42. Diagrama de despliegue

El servidor web utiliza el ORM Hibernate para gestionar los accesos a la base de datos MySQL Server.

4.5. Interfaz Android – Servidor Web

Para la comunicación entre la aplicación Android y el servidor web se ha usado la biblioteca HttpClient de Apache integrada en el SDK de Android. Su principal función es implementar métodos HTTP. La parte Android del proyecto se comunicará con la parte servidor mediante POST haciendo llamadas al servlet Listener.

El contenido de los mensajes HTTP POST que se intercambian la aplicación y el servidor web son un conjunto de instrucciones petición/respuesta en XML que se han diseñado para cada tipo de solicitud.

A continuación se describen las peticiones/respuestas de cada tipo de servicio.

Login	
Petición	**Respuesta**

`<Login>`
`  <user>usuario</user>`
`  <password>contraseña</password>`
`</Login>`
	`<Login>`
	`  <resultado>0,1</resultado>`
	`  <mensaje>Texto</mensaje>`
	`</Login>`

Lista de servicios	
Petición	**Respuesta**

`<Services>`
`  <user>usuario</user>`
`  <password>contraseña</password>`
`</Services>`
	`<Services>`
	`  <serv id="id1">Descripción 1</serv>`
	`  <serv id="id2">Descripción 2</serv>`
	`  . . .`
	`  <serv id="idN">Descripción N</serv>`
	`</Services>`

Petición servicio de taxi	
Petición	**Respuesta**

`<Taxi>`
`  <user>usuario</user>`
`  <password>contraseña</password>`
`  <fecha>dd/MM/yyyy</fecha>`
`  <hora>hh:mm</hora>`
`</Taxi>`
	`<Taxi>`
	`  <resultado>0,1</resultado>`
	`  <mensaje>Texto</mensaje>`
	`</Taxi>`

Petición servicio de limpieza	
Petición	**Respuesta**

`<Limpieza>`
`  <user>usuario</user>`
`  <password>contraseña</password>`
`  <hora>hh:mm</hora>`
`</Limpieza>`
	`<Limpieza>`
	`  <resultado>0,1</resultado>`
	`  <mensaje>Texto</mensaje>`
	`</Limpieza>`

Petición servicio de lavandería	
Petición	**Respuesta**
<Lavanderia id=”1”> <user>usuario</user> <password>contraseña</password> </Lavanderia>	<Lavanderia id=”2”> <prenda id=”1” precio=”11,11”>Descripción 1</prenda> . . . <prenda id=”N” precio=”NN,NN”>Descripción N</prenda> </Lavanderia>
<Lavanderia id=”3”> <user>usuario</user> <password>contraseña</password> <prenda id=”1”>Cantidad 1</prenda> . . . <prenda id=”N”>Cantidad N</prenda> </Lavanderia>	<Lavanderia id=”4”> <resultado>0,1</resultado> <mensaje>Texto</mensaje> </Lavanderia>

Petición servicio despertador	
Petición	**Respuesta**
<Despertador> <user>usuario</user> <password>contraseña</password> <hora>hh:mm</hora> </Despertador>	<Despertador> <resultado>0,1</resultado> <mensaje>Texto</mensaje> </Despertados>

Petición servicio de guardería	
Petición	**Respuesta**
<Guarderia> <user>usuario</user> <password>contraseña</password> <fecha>dd/MM/yyyy</fecha> <hora>hh:mm</hora> <ninos>N</ninos> </Guarderia>	<Guarderia> <resultado>0,1</resultado> <mensaje>Texto</mensaje> </Guarderia>

Petición servicio de comidas	
Petición	**Respuesta**
<Menu id=”1”> <user>usuario</user> <password>contraseña</password> </Menu>	<Menu id=”2”> <categoria id=”1”>Descripción </categoria> . . . <categoria id=”N” Descripción </categoria> </Menu>
<Menu id=”3”> <user>usuario</user> <password>contraseña</password> <categoría>id</categoria> </Menu>	<Menu id=”4”> <plato> <id>id1</id> <nombre>Texto</nombre> <descripcion>Texto</descripcion> <precio>NN,NN</precio> </plato>

	. . . </Menu>
<Menu id=”5”> <user>usuario</user> <password>contraseña</password> <plato>id</plato> <cantidad>N</cantidad> </Menu>	<Menu id=”6”> <resultado>0,1</resultado> <mensaje>Texto</mensaje> </Menu>

Petición servicio de prensa	
Petición	**Respuesta**
<Prensa id=”1”> <user>usuario</user> <password>contraseña</password> </Prensa>	<Prensa id=”2”> <periodico id=”1” precio=”11,11”>Nombre 1</periodico> . . . <periodico id=”N” precio=”NN,NN”>Nombre N</periodico> </Prensa>
<Prensa id=”3”> <user>usuario</user> <password>contraseña</password> <prensa>id</prensa> <estado>0,1</estado> </Prensa>	<Prensa id=”4”> <resultado>0,1</resultado> <mensaje>Texto</mensaje> </Prensa>

Petición servicio de cambio de divisas	
Petición	**Respuesta**
<Divisas id=”1”> <user>usuario</user> <password>contraseña</password> </Divisas>	<Divisas id=”2”> <divisa id=”1” cambio=”11,11”>Texto1</divisa> <divisa id=”N” cambio=”NN,NN”>TextoN</divisa> </Divisas>
<Divisas id=”3”> <user>usuario</user> <password>contraseña</password> <divisa id=”N”>Cantidad</divisa> </Divisas>	<Divisas id=”4”> <resultado>0,1</resultado> <mensaje>Texto</mensaje> </Divisas>

Información estática	
Petición	**Respuesta**
<Alrededores> <user>usuario</user> <password>contraseña</password> </Alrededores>	<Alrededores> <nombre name=”lugar1”>Descripcion1</nombre> . . . <nombre name=”lugarN”>DescripcionN</nombre> </Alrededores>

Petición servicio de alquiler de coches	
Petición	**Respuesta**

<Alquiler id="1"> <user>usuario</user> <password>contraseña</password> </Alquiler>	<Alquiler id="2"> <categoria id="1">Descripción 1</categoria> . . . <categoria id="N" Descripción N</categoria> </Alquiler>
<Alquiler id="3"> <user>usuario</user> <password>contraseña</password> <categoría>id</categoria> </Alquiler>	<Alquiler id="4"> <coche> <id>id1</id> <nombre>Texto</nombre> <descripcion>Texto</descripcion> <precio>NN,NN</precio> </coche> </Alquiler>
<Alquiler id="5"> <user>usuario</user> <password>contraseña</password> <coche>id</coche> <fecha>dd/MM/yyyy</fecha> <hora>hh:mm</hora> <dias>N</dias> </Alquiler>	<Alquiler id="6"> <resultado>0,1</resultado> <mensaje>Texto</mensaje> </Alquiler>

Historial	
Petición	**Respuesta**
<Historial> <user>usuario</user> <password>contraseña</password> </Historial>	<Historial> <servicio id="id1"> <nombre>texto</nombre> <fecha>dd/MM/yyyy</fecha> <hora>hh:mm</hora> <estado>Estado</estado> </servicio> </Historial>

Cancelar	
Petición	**Respuesta**
<Cancelar> <user>usuario</user> <password>contraseña</password> <id>idServicio</id> </Cancelar>	<Cancelar> <resultado>0,1</resultado> <mensaje>Texto</mensaje> </Cancelar>

4.6. Añadir nuevos servicios

Aplicación Android

Añadir un servicio a la aplicación no es muy difícil desde el punto de vista del programador. Una vez definido el nuevo servicio que se desea prestar, el programador se puede fijar en las plantillas ya creadas. Si hay alguna que se amolde, pone un identificador en la clase de menú de servicios, después deberá definir el XML de comunicación con el servidor, añadirlo en la clase de comunicación y en la clase asíncrona que se encarga de la comunicación añadir esta nueva operación.

La idea que se nos planteó a la hora de diseñar los diferentes servicios era, que si el día de mañana quisiéramos añadir nuevos servicios, se podrían reutilizar controles y estructuras diseñadas en alguno de los servicios ya implementados. Viendo los servicios ya creados, encontramos controles para gestionar fechas, horas, cantidades, menús y submenús, controles de tipo check, controles parecidos a un carro de la compra o controles con selección de opciones.

Servidor

Añadir un nuevo servicio implica realizar las siguientes acciones en la parte del servidor:

- Agregar una nueva entrada en la tabla servicios
- Escribir todo el código que procesará el nuevo servicio en un fichero llamado XmlNombreServicio alojado en el paquete tfg.beans.xml. Este fichero será cargado por el Listener cuando la aplicación móvil solicite en nuevo servicio. La clase debe tener un método llamado solicitar que será el encargado de comenzar la tramitación del servicio.

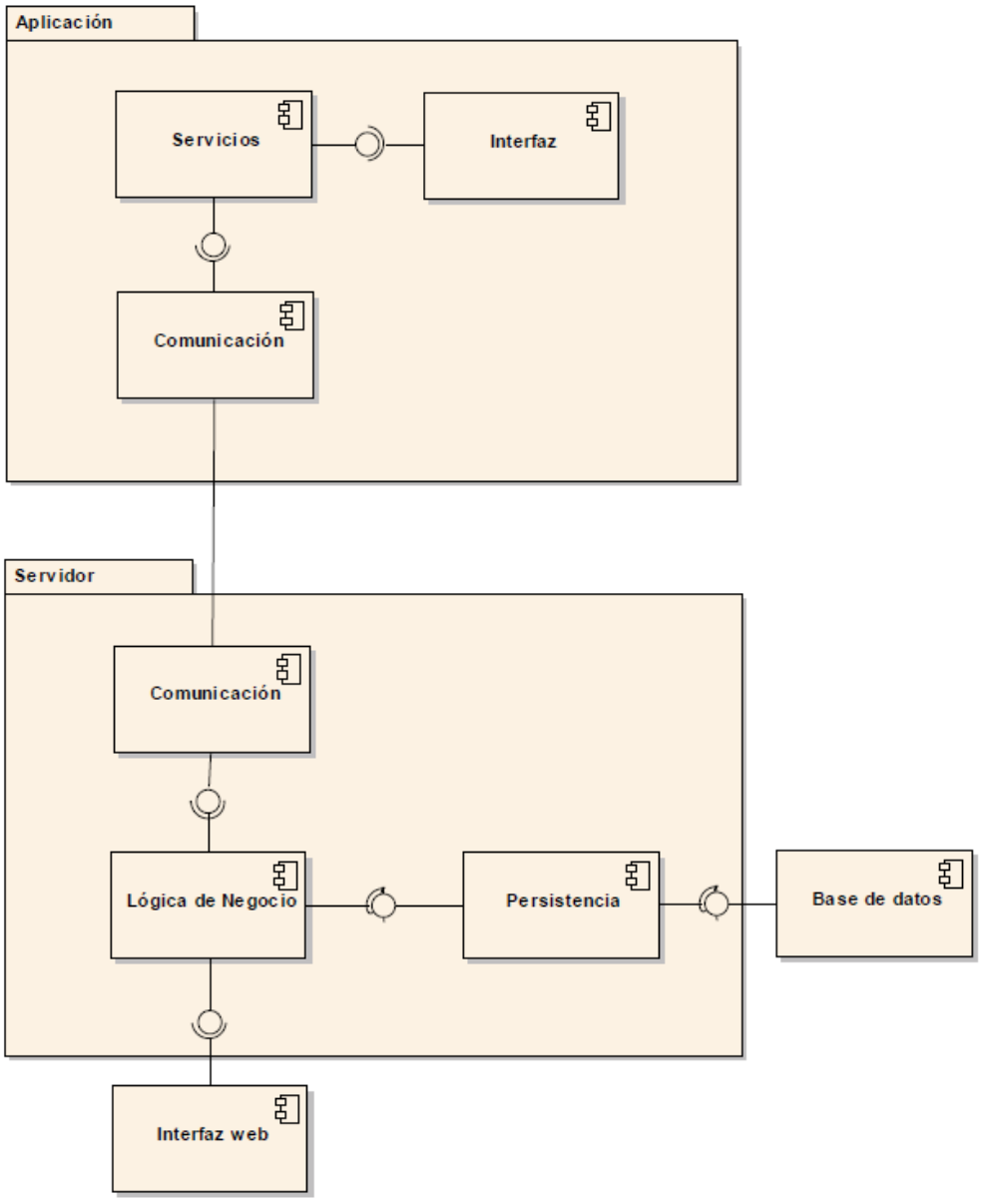
  El siguiente fragmento de código del servlet Listener permite la carga dinámica de cualquier fichero XmlNombreServicio y la ejecución del método solicitar que dará lugar a la tramitación del servicio.

```
ObjecttempClass = Class.forName("tfg.beans.xml.Xml" +
XmlComunes.getNombreServicio(xmlOrden)).newInstance();

Class claseCargada = tempClass.getClass();

Class[] argumentos = new Class[2];

argumentos[0] = Document.class;

argumentos[1] = String.class;

Methodmetodo = claseCargada.getDeclaredMethod("solicitar", argumentos);

DocumentxmlRespuesta = (Document) metodo.invoke(tempClass, xmlOrden, ruta);
```

Aplicación web

Para añadir un nuevo servicio en la aplicación web se deberá crear un jsp específico para ese servicio, y los servlets necesarios para su procesamiento.

4.7. Diagrama de componentes



5. MANUAL DE USUARIO

5.1. Aplicación Android

En la pantalla principal de la aplicación, el cliente debe de rellenar el nombre de usuario y la contraseña proporcionadas por el hotel para que pueda acceder. Tiene recordar la contraseña y el nombre de usuario para que la próxima vez no tenga que volver a introducirlos. Una vez rellenados dichos datos, pulse aceptar para poder arrancar la aplicación.

Ilustración 43. Pantalla login aplicación Android

5.1.1. Servicios primarios

En la pantalla de servicios primarios, podrá elegir tres opciones:

Ilustración 44

- Alrededores: que proporcionará información sobre museos, transportes, restaurantes externos al hotel e información sobre el hotel.
- Historial: donde se podrán ver todos los servicios solicitados, diferenciado por el estado en que se encuentren: Pendiente, representado por un color blanco, Finalizado en color verde y cancelado, ya sea por el cliente o por el hotel, en color rojo.Desde el propio historial se permitirá cancelar servicios que aun estén en estado pendiente, haciendo una pulsación larga en aquel servicio que se desee anular.Además el historial cuenta con un buscador para filtrar los resultados.

Ilustración 45

- Servicios, donde se podrá solicitar los servicios del hotel.

Ilustración 46 Servicios disponibles

5.1.2. Solicitar servicios

5.1.2.1. Alquiler de coches

El cliente podrá solicitar un alquiler de coche desde el hotel. El cliente podrá elegir el tipo de vehículo y la gama del mismo. En la pantalla de pago, podrá indicar el número de días que querrá el coche (hasta un máximo de 30 días) y la aplicación le indicará la cantidad que deberá abonar para que se pueda realizar el alquiler.

Ilustración 47- Selección categoría vehículo

Ilustración 48. Selección vehículo

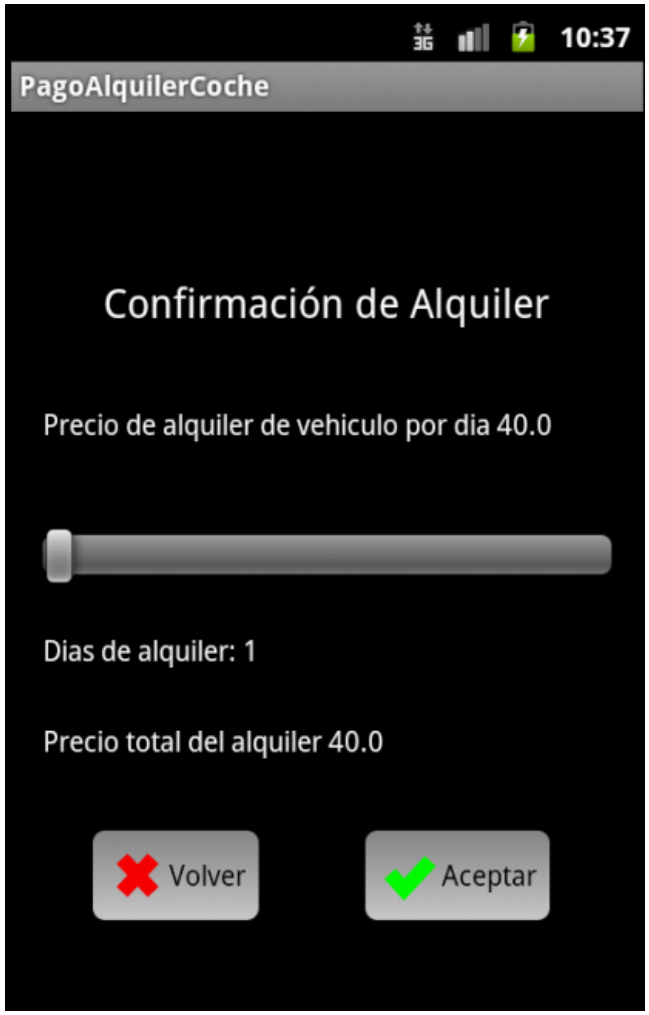


Ilustración 49. Selección días alquiler vehículo y confirmación

5.1.2.2. Cambio de divisas

El cliente podrá hacer un cambio de un tipo de moneda a otra. La aplicación mostrará al cliente a cuanto está el cambio según el cambio de moneda que elija, y cuánto dinero recibirá en la nueva moneda.

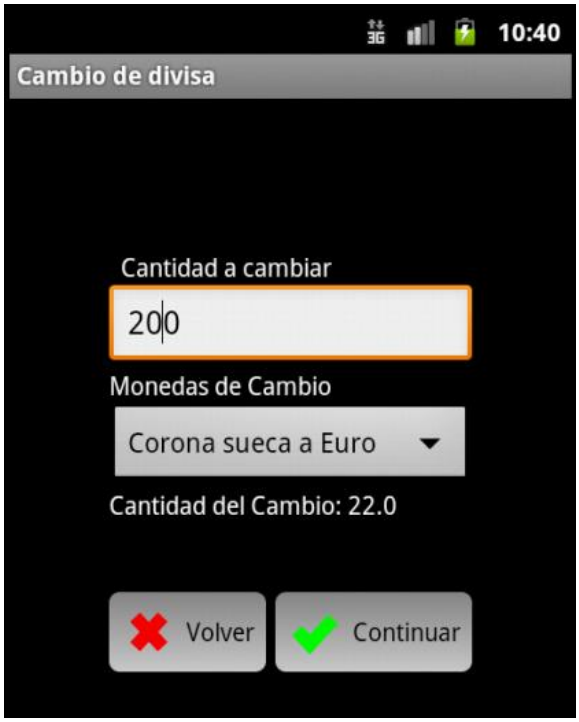


Ilustración 50. Selección de las monedas a cambiar

5.1.2.3. Guardería

El servicio de guardería permite al cliente poder reservar a una hora determinada el un servicio de guardería, donde tendrá que indicar el número de niños que tendrán que cuidar (en caso de que el hotel disponga del servicio indicado).

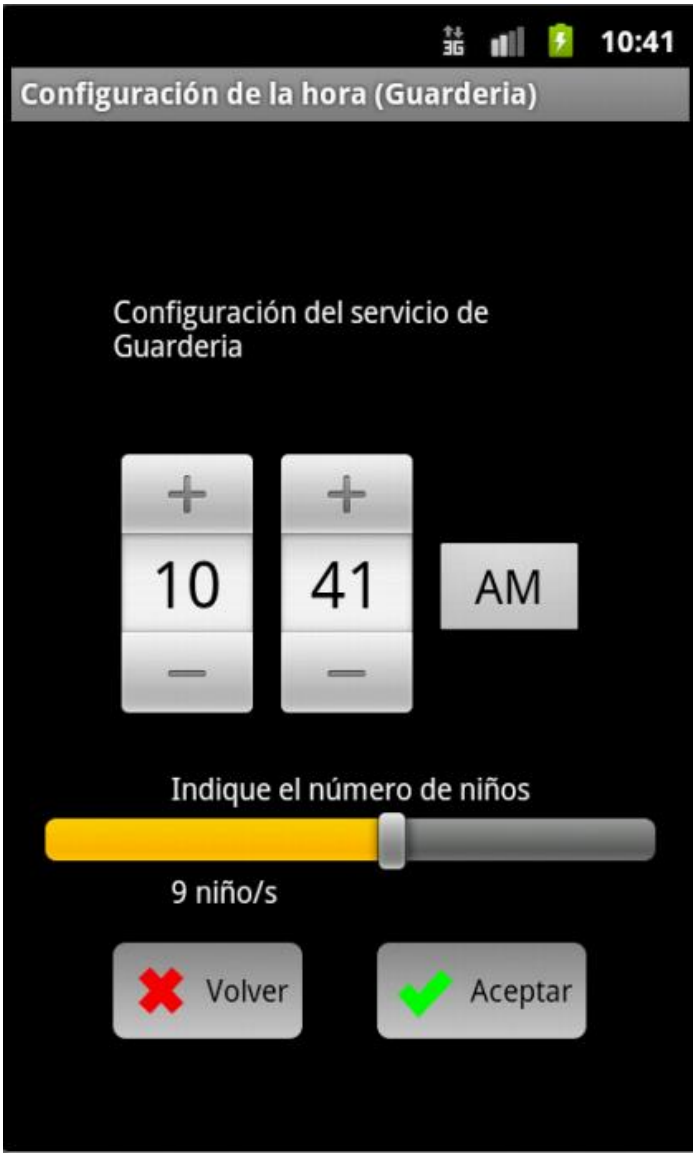


Ilustración 51. Selección servicio de guardería

5.1.2.4. Lavandería

El cliente podrá solicitar el servicio de lavandería. En la pantalla de lavandería, el cliente podrá ir añadiendo aquellas prendas que quiera que le laven al igual que el número de cada tipo de prenda. Pulsará el botón "añadir" para añadir la prenda al servicio. Una vez que el cliente haya finalizado de introducir las prendas, pulsará el botón enviar para que se pueda solicitar el servicio (en caso de que el hotel disponga del servicio indicado).

Ilustración 52. Selección de prendas

5.1.2.5. Servicio de comidas

El cliente podrá solicitar un menú del hotel tan solo pulsando la opción en el listado de servicios. A continuación, saldrá un nuevo listado con las categorías de los menús del hotel. El cliente pulsará sobre una de las categorías y saldrá una nueva ventana con todos los menús de dicha categoría. El cliente elige uno de los menús, y saldrá una nueva ventana indicando el precio de dicho menú, y permitirá al cliente indicar el número de menús que quiere del que acaba de elegir. Una vez que haya rellenado dicho campo, pulsará el botón aceptar para solicitar dicho servicio (en caso de que el hotel disponga del servicio indicado).

Ilustración 53. Selección categorías menú

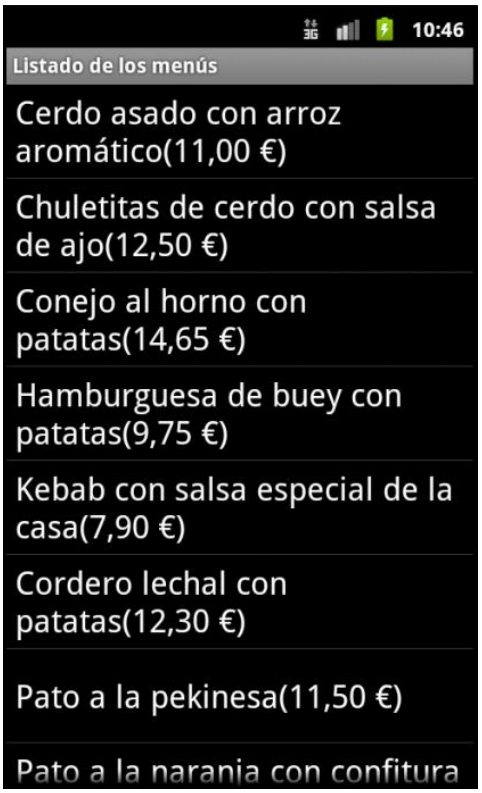


Ilustración 54. Selección platos menú

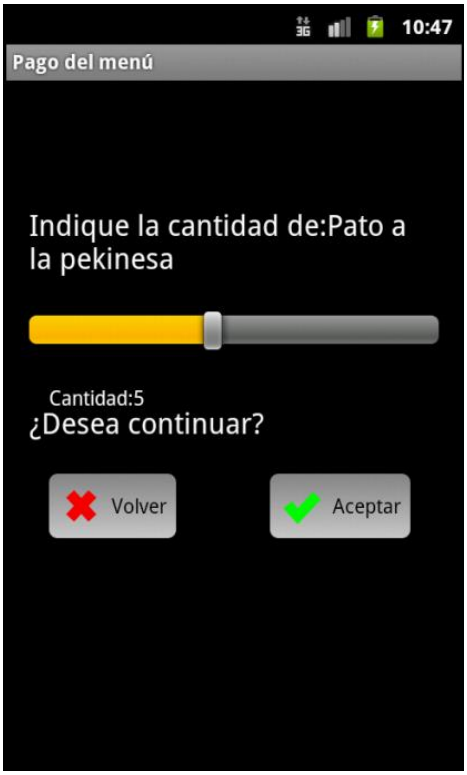


Ilustración 55. Selección número de platos menú

5.1.2.6. Servicio de Limpieza

El cliente podrá solicitar al hotel la hora a la que quiere que pasen a limpiar la habitación del cliente. El hotel dispondrá de unos horarios para la limpieza y el cliente elegirá uno y pulsará el botón aceptar si desea solicitar dicho servicio (en caso de que el hotel disponga del servicio indicado).

Ilustración 56. Selección servicio de limpieza

5.1.2.7. Servicio de prensa

El cliente podrá solicitar el servicio de prensa del hotel. Podrá elegir una hora dentro del horario disponible por el hotel, y el diario que desea que le entreguen a dicha hora. Una vez elegidos estos parámetros, el cliente pulsará sobre el botón aceptar para solicitar el servicio (en caso de que el hotel disponga del servicio indicado).

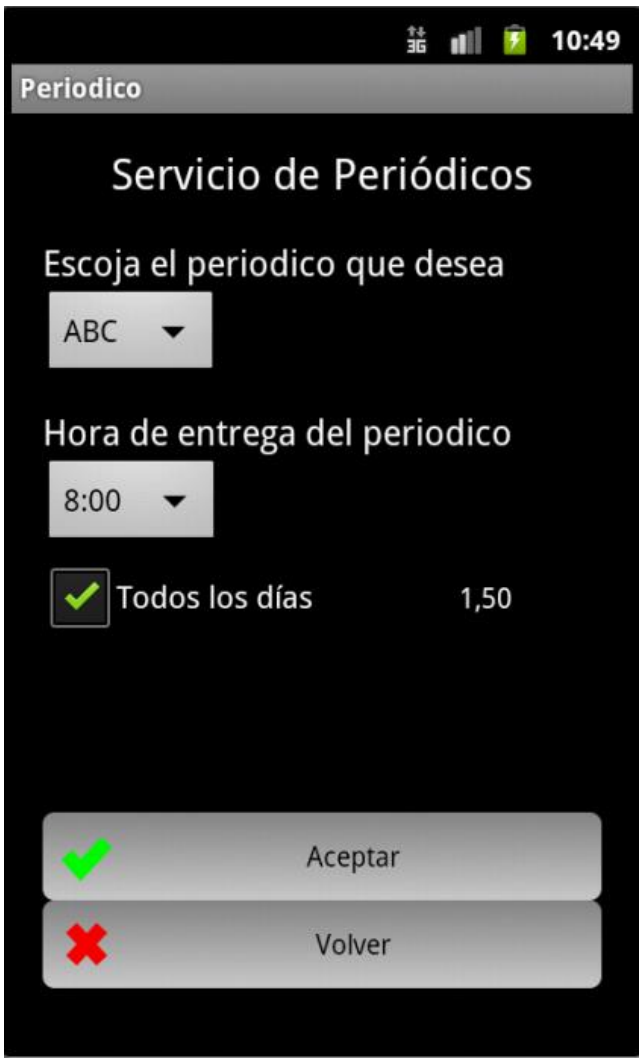


Ilustración 57. Selección servicio de prensa

5.1.2.8. Servicio de taxi

El cliente podrá solicitar un taxi que le recoja en el hotel. Para ello, el cliente elegirá la fecha y la hora a la que quiere que aparezca el taxi en la entrada del hotel y pulsará sobre el botón aceptar para solicitar dicho servicio (en caso de que el hotel disponga del servicio indicado).

Ilustración 58. Selección servicio de taxi

5.1.2.9. Servicio despertador

El cliente podrá solicitar el servicio de despertador del hotel. Para ello, el cliente seleccionará la hora a la que quiere el despertador y pulsará en el botón aceptar para solicitar dicho servicio (en caso de que el hotel disponga del servicio indicado).

Ilustración 59. Selección servicio despertador

5.2. Aplicación Web

5.2.1. Login

Desde la pantalla principal los clientes y empleados del hotel pueden acceder a la funcionalidad de la aplicación tecleando su nombre de usuario y contraseña.

Ilustración 60. Login aplicación web

5.2.2. Checkin

Desde el menú Reservas, pulsando en checkin aparecen de forma predeterminada las reservas pendientes de registra en el día actual. Usando el buscador se pueden consultar reservas en base a otros criterios.

Para realizar el checkin se pulsa sobre el número de reserva en la tabla de resultados.

Ilustración 61. Buscador reservas

A continuación se muestran los datos de la reserva y se debe asignar al cliente un número de habitación entre las disponibles según el tipo que se especificó en el momento de la reserva.

Ilustración 62. Datos reserva

Si los datos son correctos pulsando en Grabar confirmamos la reserva.

Ilustración 63. Confirmación checkin

Y el sistema genera un pdf para el cliente con el resumen de sus datos y la contraseña para que pueda solicitar los servicios del hotel a través de su móvil o de un ordenador personal conectado a interne.

Resumen Reserva

Bienvenido a nuestro Hotel, estos son los datos de su reserva:

Nº de reserva:	9
Cliente:	Maria Luisa Peralta Flores
Fecha de entrada:	03/06/2014
Fecha de salida:	04/06/2014
Nº habitación:	101
Tipo de habitación:	Individual
Forma de pago:	Pago on line web hotel

Durante su estancia puede utilizar los servicios online que ofrecemos a través de su smartphone, para ello descargue nuestra app desde este código QR, e introduzca el usuario y contraseña que le proporcionamos a continuación.

Usuario: 1633

Contraseña: kCu4yf3U

Ilustración 64. Hoja informativa para el cliente

5.2.3. Checkout

Desde el menú Reservas, pulsando en la opción Checkout entramos en la pantalla que gestiona las salidas de clientes del hotel. De manera predeterminada aparecen las salidas de la fecha actual. Usando el buscador podemos buscar salidas en base a otros criterios. Para tramitar una salida se hace click sobre el número de reserva en la tabla de resultados.

Ilustración 65. Buscador checkout

Si los datos son correctos, pulsando el Imprimir obtenemos la factura para el cliente

Ilustración 66. Pantalla confirmación checkout

HappH

Resumen servicios a facturar

Nº de reserva:	9
Cliente:	Maria Luisa Peralta Flores
Fecha de entrada:	03/06/2014
Fecha de salida:	04/06/2014
Nº habitación:	101 Individual
Importe habitación:	78,50
Forma de pago:	Pago on line web hotel

Servicio	Fecha	Importe
Menú - Habas con jamón ibérico (2)	04/06/2014	19,00
Menú - Pastel de chocolate (1)	05/06/2014	4,50

Suma Servicios	23,50
Total Factura	102,00

05/06/2014 00:10:33

Ilustración 67. Factura para el cliente

5.2.3. Solicitud de un servicio vía web

Para solicitar un servicio por parte del cliente, éste debe pinchar en el Menú Servicios, en la opción solicitar servicio. Allí se mostrará la lista de servicios disponibles. Selecciona el que desea tramitar y pulsa en Siguiente.

Ilustración 68. Solicitud de un servicio

Se introducen los datos que se solicitan para el servicio elegido y se pulsa en Siguiente.

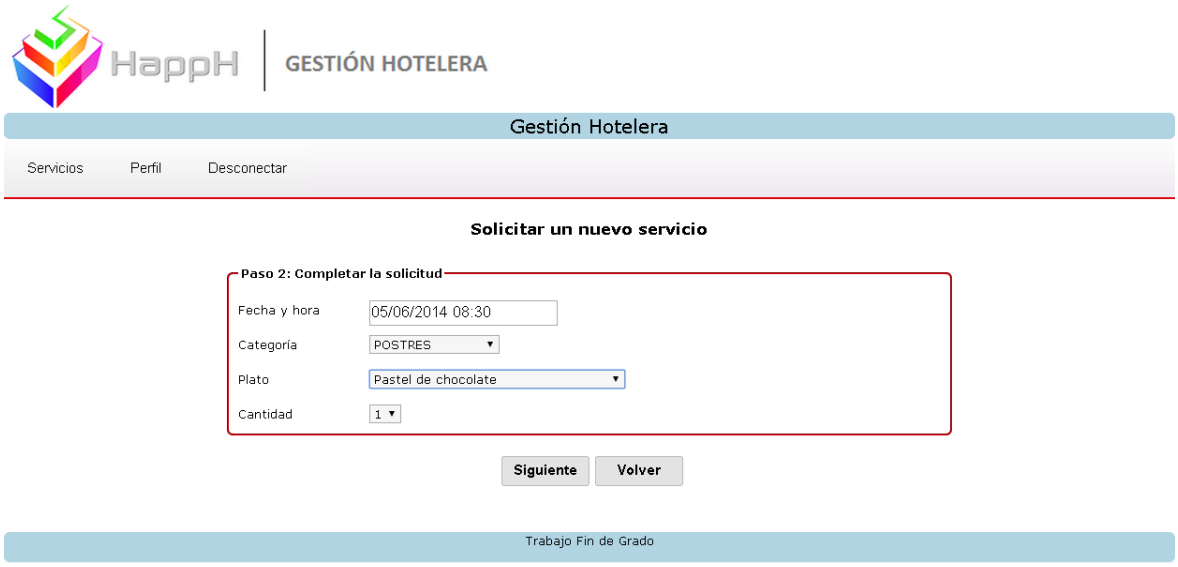


Ilustración 69. Datos del servicio solicitado

La aplicación muestra un resumen, si el cliente está conforme debe pulsar en Confirmar.

Ilustración 70. Petición de confirmación del servicio solicitado

Y por último la aplicación nos indica que el servicio ha sido solicitado.

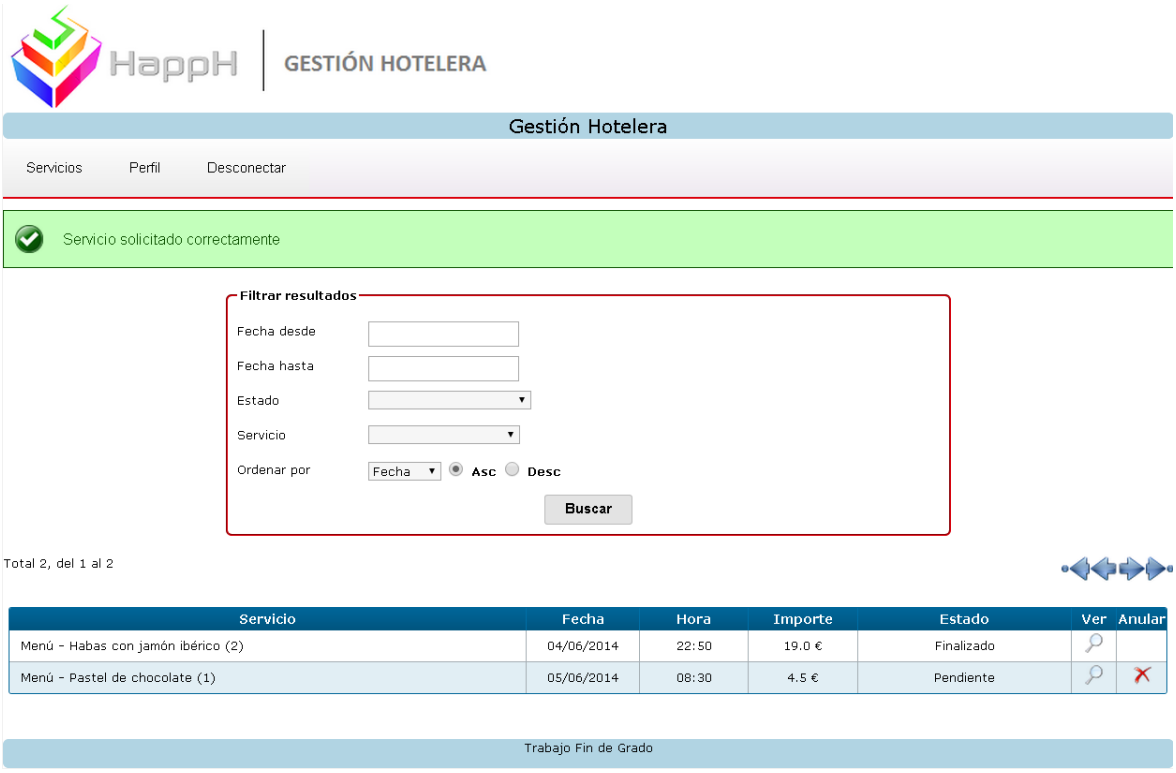


Ilustración 71. Servicio solicitado correctamente

6. RESULTADOS

En este apartado trataremos de mostrar una crítica a nuestro método de trabajo y la organización para llevar a cabo las tareas y las conclusiones que hemos sacado.

6.1. Metodología de trabajo

Al comienzo del proyecto se fijó Scrum como metodología para el desarrollo del proyecto. Esta metodología está definida principalmente por su concepción holística del desarrollo, así como un procedimiento iterativo donde se dan avances incrementales que amplían las funcionalidades implementadas o añaden nuevos requisitos, buscando siempre primar la agilidad y la prontitud para tener productos software parcial o plenamente funcionales.

Esta visión contrasta con la más tradicional y rígida, definida por fases secuenciales, la cual resulta poco flexible. Sabiendo que los requisitos y especificaciones de proyectos son proclives a introducir modificaciones, mejoras y correcciones, e incluso en ocasiones los requisitos no son conocidos en su totalidad, ésta metodología se muestra como la más acertada.

Así mismo, el número reducido de participantes en el proyecto, hacía fácil la coordinación y comunicación exigida para este tipo de desarrollos, que se basaron principalmente en las reuniones formales y, sobre todo, en la comunicación constante y casi diaria del equipo.

En las primeras reuniones se definieron los requisitos y objetivos iniciales del proyecto que finalmente sufrieron modificaciones, orientándolos hacia una mayor interactividad del usuario de la aplicación final en detrimento de características inteligentes. El total de reuniones formales ascendió a siete, aunque como se dijo anteriormente, la puesta en común de la información relativa al proyecto se hizo de manera constante durante el curso, por coincidir la mayor parte de los miembros del equipo en diversas asignaturas del Grado.

Se decidió que de los cuatro integrantes del equipo, una mitad, compuesta por Carlos Gutiérrez y Jacobo Olmedo, se dedicarían al desarrollo de la aplicación móvil para la plataforma Android, mientras que la otra, compuesta por Juan Feliu y Javier Pavón, harían lo propio con la parte del servidor y la interfaz web. A pesar de la división del trabajo se necesitó una especial coordinación en lo referente a la especificación del protocolo de comunicación entre la parte Android y el servidor web, y en las pruebas finales del sistema en su conjunto.

El seguimiento y organización del proyecto tuvo su lugar inicialmente en la plataforma Assembla (https://www.assembla.com), y a medida que se avanzaba el proyecto se extendió al uso de otras herramientas, principalmente chat y correo electrónico, así como repositorios online para el código de la aplicación.

Estas herramientas de repositorios y control de versiones fueron Git, a través de la herramienta de repositorio en Assembla, y también carpetas compartidas en Google Drive (http://drive.google.com) y Mega (http://mega.nz).

Para el desarrollo se utilizaron las herramientas Eclipse, en concreto su versión ADT (*AndroidDevelopmentToolkit*) para la aplicación Android y Netbeans para la parte web y del servidor.

6.2. Métricas

A continuación se muestra la planificación de tareas identificadas en cada una de las reuniones que se establecieron a lo largo del proyecto, su duración y los integrantes del equipo que debían llevarlas a cabo.

Como resumen de la planificación se puede destacar que para la construcción del proyecto se han necesitado 172 días de trabajo con una media de dedicación por día de unas 3 o 4 horas.

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Recursos
Primera reunión	1 día	mar 29/10/13	mar 29/10/13		Carlos;Jacobo;Javier; Juan
Casos de uso 1, 2, 3	4 días	mié 30/10/13	lun 04/11/13	1	Jacobo
Casos de uso 4, 5, 6	4 días	mié 30/10/13	lun 04/11/13	1	Juan
Casos de uso 7, 8, 9	4 días	mié 30/10/13	lun 04/11/13	1	Javier
Casos de uso 10, 11, 12	4 días	mié 30/10/13	lun 04/11/13	1	Carlos
Evaluación casos de uso	1 día	mar 05/11/13	mar 05/11/13	2;3;4;5	Carlos;Jacobo;Javier ;Juan
Construcción prototipo web	5 días	mié 06/11/13	mar 12/11/13	6	Javier;Juan
Construcción de la arquitectura de la aplicación	5 días	mié 06/11/13	mar 12/11/13	6	Carlos;Jacobo
Redefinir checkin, checkout	2 días	mié 13/11/13	jue 14/11/13	7	Juan
Creación de las plantillas base de la aplicación Android	2 días	mié 13/11/13	jue 14/11/13	7	Carlos;Jacobo
Usar Hibernate	13 días	mié 13/11/13	vie 29/11/13	7	Javier
Segunda reunión	1 día	lun 02/12/13	lun 02/12/13	9;11	
Completar servicio web	5 días	mar 03/12/13	sáb 07/12/13	12	Juan;Javier
Corrección de bugs de las plantillas Android	5 días	mar 03/12/13	lun 09/12/13	12	Carlos;Jacobo
Definir protocolo comunicación XML	8 días	lun 09/12/13	mié 18/12/13	13	Carlos;Jacobo;Javier; Juan
Tercera reunión	1 día	jue 19/12/13	jue 19/12/13	15	Carlos;Jacobo;Javier
Mejorar uso Hibernate	4 días	lun 23/12/13	jue 26/12/13	15	Javier
Avanzar protocolo XML Comunicación	3 días	lun 23/12/13	mié 25/12/13	15	Carlos;Javier

Redefinición de las clases XMLParser y XMLCreator de la parte Android, terminar de implementar el prototipo Android	6 días	jue 19/12/13	jue 26/12/13	15	Jacobo
Añadir las nuevas plantillas de los nuevos servicios Android definidos	6 días	jue 19/12/13	jue 26/12/13	15	Carlos
Web con acceso para clientes	7 días	mié 18/12/13	jue 26/12/13	15	Juan
Cuarta reunión	1 día	vie 27/12/13	vie 27/12/13	16;17;18;19;20	Carlos;Jacobo;Javier; Juan
Nuevos servicios. Casos de uso	3 días	mar 21/01/14	jue 23/01/14	21	Carlos;Jacobo;Javier; Juan
Nuevos servicios. Implantar(Web)	7 días	vie 24/01/14	lun 03/02/14	22	Juan
Probar app,añadir menú contextual, añadir aboutus,Mejora de diseño dela interfaz	7 días	vie 24/01/14	lun 03/02/14	22	Carlos
Modificar XML nuevos servicios	7 días	mar 04/02/14	mié 12/02/14	23;24	Carlos;Javier
Añadir estados a servicios	3 días	mié 05/03/14	vie 07/03/14	23	Javier
Añadir estados a servicios(Android),clase adicional para almacenar datos usuario,actividadguardería,diferenciar visualmente el historial y filtrado	3 días	mié 05/03/14	vie 07/03/14	23	Carlos;Jacobo
Modificar web estados servicios	4 días	sáb 08/03/14	mié 12/03/14	25;26	Juan
Quinta reunión	1 día	jue 13/03/14	jue 13/03/14	28	Carlos;Jacobo;Javier; Juan
Modificar servicio lavandería	2 días	mar 25/03/14	mié 26/03/14	29	Javier
Modificar web lavandería	3 días	mar 25/03/14	jue 27/03/14	29	Juan
Modificar Android lavandería	3 días	vie 14/03/14	mar 18/03/14	29	Carlos
Modificar Android Alrededores y Limpieza,arreglar bugs historial, permitir eliminar peticiones,arreglo servicio periódico	3 días	vie 14/03/14	mar 18/03/14	29	Jacobo
Modificar XML Lavandería	3 días	vie 28/03/14	mar 01/04/14	30;31;33;32	Javier;Carlos
Añadir historial (Web)	2 días	mié 02/04/14	jue 03/04/14	34	Juan;Javier
Cambiar historial en Android mediante peticiones web,no mostrar por defecto teclado en historial,definir combinación colores app,incluir en lavandería diversas opciones de prendas	8 días	mié 02/04/14	vie 11/04/14	34	Carlos;Jacobo
Pruebas integración web-Android (Comunicación)	8 días	lun 14/04/14	mié 23/04/14	35;36	Javier;Carlos
Sexta reunión	1 día	mié 23/04/14	mié 23/04/14	37	Carlos;Jacobo;Javier; Juan
Definir índice memoria	1 día	jue 24/04/14	jue 24/04/14	38	Jacobo;Juan

Pruebas del sistema	10 días	sáb 26/04/14	jue 08/05/14	38	Carlos;Jacobo;Javier; Juan
Séptima reunión	1 día	mié 14/05/14	mié 14/05/14	40	Carlos;Jacobo;Javier; Juan
Diagramas de clases	1 día	lun 19/05/14	lun 19/05/14	41	Juan
Diagramas de secuencia	1 día	lun 19/05/14	lun 19/05/14	41	Javier
Diagramas de despliegue	1 día	mar 20/05/14	mar 20/05/14	43	Javier
Elaborar memoria	12 días	mié 21/05/14	jue 05/06/14	42;43;44	Carlos;Jacobo;Javier; Juan
Entrega	1 día	vie 06/06/14	vie 06/06/14	45	Carlos;Jacobo;Javier; Juan

La parte que sin duda alguna ha sido la más compleja ha sido el diseño inicial de laarquitectura de la aplicación, que realmente llevo unos meses de trabajo, ya que no se disponía de ningún profesional de la materia (hoteles).Al principio fue complicado saber qué servicios implementar, además de que a la vez había que hacerse con la programación Android, ya que aunque todos nosotros conocíamos el lenguaje Java, la forma de programar Android no es exactamente igual, debido en gran parte al manejo de las vistas, que se implementan mediante XML, y de ciertas líneas que había que escribir en el manifiesto de la aplicación, ya que si no escribías por ejemplo los permisos de la aplicación en el manifiesto (como por ejemplo el permiso de conexión a Internet), la librería que se utilizaba en la aplicación para la comunicación con el servidor no funcionaba.

Otro de los puntos más complicados fue adaptarnos al uso de Hibernate, un framework que desconocíamos totalmente y que en un principio nos resultó complicado de entender. Posteriormente, tuvimos que adaptar el diseño original que habíamos hecho de Hibernate, para adaptarlo a los patrones que fuimos introduciendo.

6.3. Líneas de trabajo futuro

En cuanto a las líneasde trabajo futuro, el equipo ha pensado en desarrollar nuevos servicios para laaplicación, además de conseguir crear una interfaz más atractiva para elcliente. Otra de las cosas que se pueden hacer es ofrecer una plataforma multilenguaje, de tal forma que se pueda expandir geográficamente el mercado de esta aplicación.

Otro punto pendiente es el poder guardar las peticiones cuando se intenten solicitad pero no haya comunicación con el servidor, bien por un problema de éste o por estar situados en alguna zona con poca cobertura telefónica, para que en cuanto se restablezca la red se puedan mandar todas las peticiones que el usuario haya solicitado.

También que este sistema se pueda extender a plataformas IOS y Windows Phone. En el caso de IOS, requeriríamos un ordenador de Apple (o una máquina virtual) además de un entorno de programación como el programa XCode de Apple y aprender a programar en "objetive-c". En el caso de Windows pone, para poder programar estos móviles se debe saber los lenguajes de programación "C#" y "Visual Basic", además de disponer una plataforma de desarrollo que soporte estos lenguajes.

Otra de las ideas planteadas como líneas de trabajo futuro es la de añadir en la aplicación Android un widget que sirva para poder visualizar las peticiones que ha hecho el cliente en el hotel. De tal forma sabrá en todo momento las peticiones canceladas, las peticiones que ya se han realizado y las que están pendientes.

Por último, otra idea que podría ser muy significativa a la hora de mejorar esta plataforma es la de poder crear un programa que permita añadir o eliminar servicios de una forma sencilla, de tal forma que esto pasa a desarrollarse de forma automática, donde el administrador podrá añadir o eliminar los servicios que considere necesarios.

Aunque cómo y cuándo se desarrollarán estas ideas dependerá de hacia dónde vayan las carreras profesionales de cada integrante del equipo y si algún día volverán a juntarse como en este Trabajo Fin de Grado.

6.4. Cumplimiento de objetivos

Sobre el cumplimiento de los objetivos del proyecto, podemos decir que gran parte de ellos se han sido cumplidos. Sin embargo, hay ciertos aspectos que no han podido llevarse a cabo debido a que ha habido ciertos problemas a la hora de poner en práctica la metodología que habíamos pensado para poder trabajar y poder así cumplir todos los objetivos. El hecho de tener más asignaturas en el curso y la falta de disponibilidad de compañeros del proyecto debido a que trabajaban, tuvieron cierta repercusión a la hora de cumplir los objetivos del proyecto. Sin embargo, los objetivos base que pusimos sí que fueron cumplidos, además de otros muchos objetivos adicionales que le daban más cuerpo al proyecto, aunque nos hubiese gustado disponer de más tiempo para poder terminar los objetivos restantes

Si tenemos en cuenta el prototipo de la aplicación que teníamos ideado y la aplicación que vemos ahora, podemos decir sin duda alguna que la aplicación que tenemos al final posee más funcionalidades y está más adaptada al mercado que la aplicación base que teníamos pensada.

Otro de los objetivos que queríamos era que fuese lo suficientemente sencilla para que los usuarios que utilizasen la aplicación no tuviesen problema alguno a la hora de poder pedir un servicio, lo cual pensamos que lo hemos conseguido con creces.

Otro de los objetivos que posteriormente pusimos para la aplicación era el estado de las peticiones de los servicios. Estaba bien eso de pedir servicios, pero, ¿Y si el servicio deja de estar disponible por una cuestión externa a HappH? ¿Y si el usuario se ha equivocado

a la hora de hacer una petición de un taxi, pensando que escogió una hora cuando era otra?, ¿Y si hubiese algún problema sobre pedidos que tiene que pagar el usuario cuando por cualquier casual le ha sido imposible disfrutar de él? Pues bien, utilizando este sistema permite ver el estado de todos los servicios que ha pedido el usuario además de que no haya problemas con dichas peticiones, ya que aparecen en color verde las que ya se han disfrutado y se dispone de lo mismo en el servidor.

En conclusión, pensamos que la aplicación es útil para poder realizar las peticiones de servicios de los clientes de un hotel, sin embargo se ha echado en falta un profesional en el tema hotelero que pudiese aconsejarnos y darnos más servicios para poder añadir a la aplicación.

Esta última parte es posible que pueda afectar negativamente a la aplicación, ya que como no somos profesionales en este tema, desconocemos muchos de los servicios que se pueden llevar a cabo en un hotel. Es por eso que lo hemos desarrollado como plantillas para que se pueda en un futuro añadir más funcionalidades.

6.5. Conclusiones

Desde el punto de vista de negocio, la aplicación muestra un ejemplo de lo que podría ser un sistema de petición de servicios para hoteles, sirviendo de base para que se puedan hacer mejores versiones en el futuro e incluir nuevas funcionalidades al mismo, haciendo que la aplicación se vaya enriqueciendo y haciéndola cada vez más funcional. Como se ha dicho en el punto anterior, hubiera sido mejor disponer de ayuda en el terreno de hoteles para que la aplicación fuera mucho más acertada. Sin embargo, estamos muy satisfechos con el trabajo que se ha realizado, ya que se ha conseguido mucho empezando desde cero, hay que tener en cuenta que anteriormente no habíamos desarrollado un proyecto de este calibre, además de no haber utilizado la plataforma Android.

En cuanto a la metodología de trabajo, nos hubiera gustado que sehubiese respetado de forma más rigurosa, ya que por motivos varios por parte de los miembros del grupo, se hacía complicado hacer reuniones físicas, por lo que al final la solución que veíamos para poder paliar este problema era la de hacer reuniones virtuales.

Por último, recalcar que a pesar de los muchos problemas que acarrea el trabajo en equipo, hemos sabido lidiarlos lo mejor posible y la verdad que estamos muy satisfechos con lo que finalmente se ha conseguido y se presenta en este documento.

Este trabajonosha enseñado que tenemos capacidad para afrontar los proyectos que nos pongan (independientemente de si conocemos o no la plataforma en la que trabajamos), y además nos ha mostrado lo que no tenemos que hacer en proyectos futuros. Uno de los puntos clave que tendremos que acuñar para próximos proyectos es la de utilizar herramientas tales como Assembla o repositorios de código de forma mucho más activa, porque hemos podido ver que cuando no se utilizan estas

herramientas de forma correcta, ponerse de acuerdo y distribuir las tareas entre los miembros del grupo puede llegar a ser muy tedioso.

6.6. ¿Qué hemos aprendido durante el desarrollo del proyecto?

Lo primero que hemos aprendido, y que creemos que es uno de los factores más importantes, es el de trabajar en grupo. Estamos acostumbrados a trabajar en grupos de dos personas en ciertas asignaturas, pero a diferencia de estas, en el Trabajo Fin de Grado éramos cuatro. Además, en las asignaturas de la carrera nos guiaban los profesores para poder llevar a cabo las prácticas, mientras que en el proyecto todo lo que hemos aprendido sobre servidores y Android hemos tenido que trabajarlo por nuestro lado, lo cual es bastante más costoso.

En cuanto a cosas que nos hubiera gustado hacer, una de ellas hubiese sido añadir más funcionalidades al proyecto, que pueda prestar más servicios para que fuera más completo (que ese era uno de nuestros objetivos adicionales que por tema de tiempo no pudimos llevar a cabo). También nos hubiese gustado añadir mayor facilidad de personalización a la aplicación a nivel de usuario, es decir, que a través de la aplicación se pudiesen configurar ciertos aspectos tales como los colores, o el tipo de letra (para gente que tiene dificultades visuales y requieren ver las letras de nuestra aplicación más grandes).

En cuanto a cosas útiles que hayamos hecho durante el desarrollo del proyecto, podemos destacar la división de los componentes en dos grupos, uno para el desarrollo web y para el desarrollo Android, de tal forma que pudimos así especializarnos en un área y abarcarla con mayor profesionalidad.

Destacar también el aprendizaje de Hibernate como ORM, que al principio supuso una gran dificultad de configuración, y entendimiento de cómo funciona y trabaja, pero que luego desembocó en una gran facilidad para trabajar con tablas del modelo relacional desde un modelo de programación orientado a objetos.

6.7. Descripción del trabajo individual

Carlos Gutiérrez Hernández-Gil

Antes de iniciar el desarrollo de la aplicación Android, Carlos ha tenido que documentarse y estudiar el funcionamiento de Android, además de instalar los plugins necesarios para poder realizar el desarrollo del programa en la plataforma Eclipse

Dentro del desarrollo de HappH, se ha ocupado de la parte de la aplicación desarrollada en Android junto a Jacobo. Se ha encargado de definir los servicios que iba a prestar la aplicación, además de diseñar la arquitectura de la aplicación junto a Jacobo de tal

forma que estuviese diseñada mediante plantillas para que posteriormente puedan añadirse nuevos servicios de forma sencilla.

De tal forma que según los datos que requieran servicios nuevos que se decidan añadir en la plataforma, se asemeje a una de las interfaces que previamente se ha diseñado con el propósito de que se pueda añadir el servicio sin apenas tocar el código de la aplicación

También se ha encargado junto con su compañero Javier, de definir los mensajes que se iban a intercambiar entre la aplicación y el servidor que presta los servicios a la aplicación. Para ello, se ha definido en XML unas plantillas de mensajes según los servicios de tal forma que sea fácil de extender en el caso en el que se desee añadir nuevos servicios o funcionalidades a la aplicación y requiera de comunicación con el servidor.

Además, también se ha encargado de elaborar otra aplicación cuyo cometido era el de probar a enviar múltiples peticiones al servidor de forma simultánea, para poder probar el rendimiento del servidor, viendo cuánto tarda en contestar a las peticiones y ver si el proyecto cumplía las expectativas teniendo en cuenta que el volumen de clientes de un hotel puede llegar a ser alto, por lo que es muy importante tener en cuenta el factor de múltiples envíos para ver cómo reacciona el servidor.

Carlos ha diseñado la parte de la interfaz gráfica de usuario, diseñándola de tal forma que sea fácil de comprender y que no exija demasiados pasos para poder pedir un servicio. Dado que es una aplicación que está orientada a unos supuestos clientes que lo que desean es tener las cosas de forma rápida y eficiente, se vio que cargar con demasiados objetos en las interfaces gráficas podía llegar a ser incluso contraproducente, ya que la finalidad de esta aplicación es la de ofrecer un servicio a un cliente, una comodidad a la hora de utilizarlo y que no necesite una gran cantidad de instrucciones para poder pedir dicho servicio, ya que en esta clase de aplicaciones se busca que todo sea lo más rápido posible y teniendo en cuenta que probablemente el cliente que utilice nuestra aplicación no va a desear tener que leer un manual de instrucciones o hacer un tutorial para poder realizar las peticiones dentro del hotel.

También se ha encargado de la corrección de errores que ha ido teniendo el proyecto conforme se iban sacando nuevas versiones con nueva funcionalidad. A medida que la cantidad de servicios aumentaba, ya que había nuevas ocurrencias de servicios para añadir a la aplicación y al servidor, el número de fallos que tenía la aplicación era cada vez más grande, lo cual obligó a dedicar cierto tiempo a realizar tareas de prueba para ver que todo funcionaba de forma correcta

Donde más hubo problemas fue en la parte de los XML (comunicación). A pesar de que se definió la estructura de estos mensajes que permitirían comunicar a la aplicación con el servidor, se tuvo la problemática de que la cantidad de mensajes que se hicieron para la comunicación era relativamente alto, por lo que un error a la hora de escribir una etiqueta del XML hacia que dejase de funcionar correctamente ese servicio, y además

como había muchos de los mensajes que mandaban una cantidad de datos relativamente altos, muchas veces el problema no venía de haber etiquetado mal un mensaje sino que el contenido de uno de los datos que se enviaban no estaba bien escrito, por lo que obligaba a ir paso a paso depurando el mensaje e ir viendo el contenido del mismo.

A parte, también se ha encargado de parte de la documentación del proyecto. Una vez que se ha ideado los servicios que se deseaban añadir en la aplicación, Carlos ha ido añadiendo gran parte de estas funcionalidades a los casos de uso, documentándolos según los estándares que se utilizan a la hora de realizar un desarrollo software. Al principio se decidió realizar un proyecto con sistemas inteligentes que iba a ser simulada mediante la plataforma ubiksim, por lo que se tuvo que idear los casos de uso dependiendo de este hecho, pero posteriormente se decidió quitar la parte del sistema inteligente ya que se vio que el cliente lo que buscaba era tener pleno dominio de las funcionalidades de la aplicación, por lo que fue necesario hacer una reestructuración de los casos de uso que le tocaron, además de hacer un cambio en la arquitectura de la aplicación que ya se había comenzado a implementar.

Otra de las contribuciones que tuvo fue la de idear la aplicación de tal forma que fuese personalizable, ya que se propuso que la aplicación tuviese dos colores principales que serían los colores corporativos del hotel al que se le vendiese la aplicación. De esta forma la aplicación HappH podría tener múltiples facetas según los colores corporativos que hayan decidido añadirle.

Juan Carlos Feliu Ladaria

Juan Carlos junto con Javier se ha encargado de la parte del servidor, así como de la aplicación web.

El primer paso, tras definir junto con el resto de participantes del grupo los casos de uso, fue la definición de la base de datos y de la estructura de la aplicación web.

Diseñaron la aplicación web en principio para el administrador del sistema y los empleados, con el objetivo de que existieran dos roles bien diferenciados, cada uno de ellos con unas características diferentes. La idea que tuvimos fue que el administrador fuese la persona encargada de gestionar la base de datos a través de la interfaz web, dando de alta, eliminando y modificando las diferentes tablas, como por ejemplo los empleados. Por su parte, el rol de empleado sería el encargado de hacer los checkin y checkout de los clientes y el de poder ver los servicios que se habían solicitado.

Durante el transcurso de la aplicación, cuando íbamos teniendo más registros, nos dimos cuenta de la necesidad de tener más controles para poder gestionarlos. Así surgió la idea de crear y configurar todos los controles actuales en los diferentes módulos.

El primer control lo diseñamos para el limitar el número de registros que se muestran en una página, en su parte de resultados, para evitar que se mostrasen en cadena todos los posibles resultados. Junto con este control, fue necesario añadir controles para desplazarse por estos resultados: ir a los primeros, ir a los siguientes, ir a los anteriores o ir a los últimos.

También se diseñó un control para filtrar los resultados, poder seleccionar por ejemplo aquellos empleados que estuviesen activos en el sistema, aquellas reservas que ya hubiesen hecho el check in, etéctera.

Otra de los cometidos fue el gestionar la integridad referencial de los datos. Como permitíamos al administrador y empleados poder eliminar registros, teníamos que comprobar que la eliminación no afectase a ninguna operativa, ni se eliminasen registros que podríamos usar para el histórico.

Posteriormente, con el avance de la aplicación, surgió la idea de hacer una aplicación web exclusiva para clientes, donde el cliente pudiese solicitar servicios, cancelar aquellos servicios que no estuviesen ya confirmados, y ver el historial de aquellos servicios que ya había solicitado. Fue Juan Carlos quien se encargó de toda esta aplicación.

Más adelante, la aplicación web para el administrador también contaría con módulos para la administración del resto de tablas que afectan directamente a la aplicación móvil. Así, se diseñó un sistema para que el administrador de la aplicación pueda cambiar por ejemplo las categorías de los menús que se ofrece, junto con los platos de cada categoría, así como su precio. Análogamente se puede configurar lo mismo con los vehículos y sus respectivas categorías.

Junto con Javier, ambos diseñaron y mantuvieron durante todo el proyecto la base de datos.

Por último, Juan Carlos fue el encargado de montar y mantener el servidor en su domicilio y configurarlo para conseguir sostener la aplicación. En un principio contó con un ordenador con recursos muy limitados, pero que para empezar a probar la comunicación entre la aplicación móvil y el servidor nos venía bien.

Pero según iba creciendo la aplicación, este equipo era escaso, ocasionándonos en muchas ocasiones pérdida de tiempo por reinicios y bloqueos del equipo. Consiguió hacerse con una máquina más potente, y se volvió a configurar desde cero todo el servidor, para tener a día de hoy una aplicación totalmente estable. Para facilitar al resto del grupo el acceso al servidor y no tener que depender de Juan Carlos para realizar cualquier tarea, se configuró el equipo para permitir el acceso remoto a él.

Jacobo Olmedo Martín

Jacobo, junto con Carlos, se ha encargado de la parte de la aplicación Android.

En primer lugar definió, junto al resto de miembros del equipo, los casos de uso que se consideraron a implementar en la aplicación.

Se encargó también de la gestión del sistema de control de versiones a través de la plataforma Assembla, utilizando la herramienta Git, y también se encargó de recopilar en forma de actas las ideas, comentarios y propuestas consideradas en las frecuentes reuniones, así como las líneas futuras de trabajo, durante el desarrollo del proyecto, incluyéndolas en una wiki creada al efecto en la mencionada plataforma.

Diseñó, junto a Carlos, la arquitectura a implementar en la aplicación Android, previendo una fácil extensibilidad para introducir futuribles servicios a prestar.

También fue el encargado de desarrollar el aspecto de la comunicación entre la aplicación Android y el servidor, realizando las primeras pruebas al respecto, y diseñando la jerarquía de clases implicadas en ese proceso, que implementó la comunicación de forma asíncrona.

También posibilitó la inclusión de indicadores de progreso durante las frecuentes comunicaciones cliente-servidor usando la manera oficialmente recomendada de separarlas del hilo principal de ejecución, lo que influyó considerablemente en la estructura de clases involucradas en ese proceso, para implementarlo de forma asíncrona. Esto tuvo repercusión sobre la estabilidad de la aplicación y permitió además su uso en dispositivos Android modernos, dado que desde la versión 4.0 en adelante se impide la ejecución de tareas costosas en tiempo en el hilo principal de la aplicación.

Así mismo se encargó junto con Carlos del diseño de la interfaz de usuario, adaptándola a dispositivos de pantalla pequeña. También diseñó los propios iconos de la aplicación

Android, utilizando para ello la herramienta de diseño gráfico vectorial Inkscape, priorizando la sencillez de uso para el usuario.

En relación con lo anterior, propuso la idea de brindar la posibilidad de configurar la interfaz de usuario, para adaptarla a la imagen corporativa personalizada para cada hotel.

Realizó pruebas al software desarrollado, documentándolas convenientemente, facilitando al equipo del servidor los ficheros *.apk para la instalación de la aplicación Android, para que pudieran así mismo probarla.

También intervino en la presente memoria a fin de documentar los aspectos requeridos del proyecto.

Javier Pavón Núñez

Javier, al igual que Juan Carlos, ha desarrollado la parte servidor del proyecto.

Después de identificar los requisitos iniciales y participar en el diseño de los casos de uso, empezó a trabajar en el diseño de la base de datos.

En una primera fase hizo la versión inicial de la aplicación web, el sistema en sus orígenes permitía hacer el proceso de checkin y checkout a los recepcionistas del hotel. Esta primera versión tenía un buscador de las entradas y salidas de clientes en el hotel. Utilizando el módulo JasperReports se generaban dos PDF. El primero se generaba al completar el checkin del cliente en el hotel. El documento contenía los datos identificativos del cliente, sus datos de reserva, habitación, claves para los servicios del hotel y código QR para descargar la aplicación móvil. El documento PDF del checkout era un resumen a modo de factura que incluía el importe de la estancia en el hotel y todos aquellos servicios que el cliente podría haber pedido durante su alojamiento.

Tras enseñar la primera versión de la aplicación, se ocupó de utilizar Hibernate en el proyecto en lugar de JDBC.

El uso de Hibernate en el proyecto ocasionó muchos problemas, primero por el desconocimiento de su configuración, ya que era la primera vez que tanto Javier, como Juan Carlos, utilizaban un framework de mapeo objeto-relacional. En segundo lugar por el cambio en el lenguaje de consultas. Hibernate utiliza un dialecto llamado Hibernate Query Language (HQL), que aunque tiene similitudes con SQL, tiene otras muchas diferentes que hicieron que la curva de aprendizaje y adaptación fuera pronunciada al comienzo.

Por último, y no por ello menos importantes, la gestión y uso de las sesiones y transacciones en Hibernate fueron otro gran foco de problemas. Al saber crearlas y cerrarlas adecuadamente se unió el problema del “Lazy loading” o carga perezosa, que en entornos web era más complicado de gestionar. Tras muchas consultas por diversas

webs se pudo solventar haciendo uso de filtros JEE, para abrir la sesión en un servlet y cerrarla en el jsp que mostrará al usuario el resultado.

Una vez terminada la configuración de Hibernate y aprendido su uso y puesta en práctica, Juan Carlos pasó a encargarse del desarrollo de la parte web y Javier de la implementación de los servicios identificados en los casos de uso, de la comunicación con la aplicación Android y la gestión de todas sus peticiones.

Javier junto con Carlos, decidieron que la manera de comunicar ambas aplicaciones sería a través de la clase HttpClient desde Android llamando a un servlet del servidor que estaría a la escucha de peticiones, tramitaría la operación solicitada y devolvería el resultado de la operación.

En colaboración con Carlos se definieron las instrucciones para la comunicación en formato XML. Este proceso fue bastante iterativo ya que iban surgiendo características no contempladas inicialmente y la concepción de los servicios se fue modificando a medida que avanzaba el proyecto.

El desarrollo del servlet Listener, utilizado por la aplicación móvil para comunicarse con el servidor, debería poder llamar a cualquier servicio de una forma sencilla y permitir a la vez incorporar nuevos servicios al hotel sin tener que modificar el código del Listener para no afectar a su interfaz y por ende a la aplicación Android.

Para ello se hizo uso de la carga dinámica de clases. Todas las clases que implementaban el tratamiento de un servicio comenzarían su nombre con el mismo prefijo, y todas tendrían el mismo nombre de método para invocarlas. Estas dos características posibilitaban la rápida adicción de nuevos servicios sin alterar el código del servlet de comunicación y sin afectar a su invocación desde la parte móvil del proyecto.

La última etapa antes de comenzar la elaboración de la presente memoria, fue el realizar junto con Juan Carlos una batería de pruebas a la aplicación móvil para comprobar su correcto funcionamiento, detectar errores, corregirlos si afectaban a la parte del servidor y comprobar de nuevo el funcionamiento tras los cambios. También estaba implícito el corregir los fallos detectados por los integrantes de la aplicación Android cuando probaban la interfaz web con su conjunto de pruebas.

BIBLIOGRAFÍA

- Jerome (J.F.) DiMarzio: Android, a programmers guide, McGraw Hill, 2008.
- Mark L. Murphy: Beginning Android, Apress , 2009.
- SayedHashimi, SatyaKomatineni, Dave MacLean: Pro Android 2, Apress, 2010.
- Chris Haseman: Android Essentials, Apress, 2008.
- Rick Rogers, John Lombardo, ZigurdMednieks, Blake Meike: Android application development, O'Reilly, 2009.
- Jeff Linwood, Dave Minter: Hibernate for beginners, Apress, 2010.
- SubrahmanyamAllamaraju, CedricBeust: Programación Java Server con J2EE Edición 1.3: Anaya Multimedia, 2004

Referencias web

http://stackoverflow.com/
http://developer.android.com/
http://cursohibernate.es/
http://www.chuidiang.com/
http://jqueryui.com/datepicker/
http://codejavu.blogspot.com.es/
http://achecker.ca/checker/index.php
http://community.jaspersoft.com/